21世纪高等学校实用软件工程教育规划教材

J2EE Web核心技术

——XHTML与XML应用开发

杨少波　主编

清华大学出版社

21世纪高等学校实用软件工程教育规划教材

J2EE Web核心技术
——XHTML与XML应用开发

杨少波　主编

清华大学出版社
北京

内容简介

本书继续沿用已经出版的"J2EE项目实训"、"J2EE课程设计"系列教材的技术风格，选择目前比较热门的Web 2.0技术、主流的J2EE平台中的各种核心技术，并结合项目开发的具体实例进行详细和深入介绍。

本书共9章，内容分为3大部分。前4章主要涉及应用"XHTML+CSS+Div+JS"构建Web网站必须掌握的HTML/XHTML、CSS、Div和JavaScript语言等方面的技术及应用；第5、6章的内容主要涉及XML技术及文档类型定义(Document Type Definition，DTD)、XML Schema技术及应用等XML语法相关内容；第7、8和9章的内容属于XML的转换和解析方面的技术内容，主要包括XML XSLT技术及应用、XML XPath技术及应用、XML DOM技术及应用等。

本系列教材适合作为承担国家技能型紧缺人才培养培训工程的高等职业院校及示范性软件学院的计算机应用与软件工程专业的J2EE技术平台下的应用开发类课程的教学和学习教材，也可作为自学和急需了解J2EE技术平台的软件项目开发和实现的相关技术及知识的技术人员的参考书。当然也适合作为各类职业技能培训机构的J2EE应用开发类培训课程的教材。

图书在版编目(CIP)数据

J2EE Web核心技术：XHTML与XML应用开发/杨少波主编.—北京：清华大学出版社，2011.1
(21世纪高等学校实用软件工程教育规划教材)
ISBN 978-7-302-23872-0

Ⅰ. ①J…　Ⅱ. ①杨…　Ⅲ. ①JAVA语言－程序设计－高等学校－教材　Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第181044号

责任编辑：高买花
责任校对：梁　毅
责任印制：

出版发行：清华大学出版社　　**地　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　销：全国新华书店
开　本：185×260　**印　张**：23.25　**字　数**：576千字
版　次：2011年1月第1版　**印　次**：2011年1月第1次印刷
印　数：1～　000
定　价：　.00元

产品编号：

PREFACE 序言

为了保证我国软件人才的培养，教育部于2001年发出了《教育部关于试办示范性软件学院的通知》，迄今为止全国已经拥有36家示范性软件学院，在软件人才培养方面开辟出一条崭新且有效的道路，为国家软件产业的迅猛发展提供了人力资源保证。

尽管近年来我国在软件人才的教育、培养方面取得了显著的成就，累计培养软件工程专业毕业生6万余人，人才数量与质量年年提高。但目前我国的软件教育也还存在许多问题，例如优秀软件工程专业教材匮乏，教材的理论、技术明显落后。这主要是由于我国学校开设软件工程专业的时间相对较晚，目前教学理念、方向、手段和教学内容等尚未统一；兼之软件业发展日新月异，而新理论与新技术从产生到由专家学者著书论述，再到编写教材、出版，最后到学校讲授往往已经滞后了好几年了。这是目前我国软件工程教育急需解决的一个难题。

有鉴于此，为适应我国经济结构战略性调整的要求和软件产业发展对人才的迫切需求，实现我国软件人才培养的跨越式发展，北京交通大学国家示范性软件学院与清华大学出版社合作，决定推出《21世纪高等学校实用软件工程教育规划教材》系列丛书，以先进的教学理念和教学方法，最新的实用软件技术提高软件专业的教学水平和教材质量，填补国内高等院校软件专业教材的空白，引导和规范国内高等院校软件专业教育的方向。

北京交通大学国家示范性软件学院成立于2003年。作为国家重要的软件人才培养基地，成立5年多来，在管理体制、运行机制、教育思想与理念、人才培养方案与课程体系、教学模式与方法、产学研合作等领域大胆创新，探索出一条有效地培养"国际化、工业化、高层次、复合型"软件人才的办学之路，推出了"2+1+1"的人才培养模式。在软件工程专业课程体系建设、专业课程教学、实训实习等方面取得了丰富的经验。

本系列教材是针对当前高等教育改革与发展的形势，以社会对人才的需求为导向，主要以培养高素质应用型软件人才为目标，立足软件工程专业课程体系完善与教材规范。本系列教材以北京交通大学国家示范性软件学院多年教学经验为基础，听取多方面专家的意见，主要结合软件企业的实际需要，由具有丰富行业背景的企业教师执笔完成。主要贯彻"做中学"的教育理念，注重案例体验式教学，注重学生实际能力的培养，供普通高等院校软件工程专业学生参考使用。

由于主观或客观的诸多限制，丛书难免有不尽如人意之处。敬请有志于从事软件工程教育的广大专家、学者、同仁、读者以及软件行业的杰出人士一道，相互切磋探讨，以便共同促进我国软件业的发展和繁荣。

编委会

2008 年 2 月

FOREWORD

前言

1. 为什么要提出编写 J2EE Web 核心技术系列教材

（1）高校教师希望提供“系列化”的教学支持和帮助

由于高校在校学生在四年的学习过程中会处于不同的知识层次和技术应用层次，而不同层次的学校老师和学生对教材的“深”、“浅”也有不同的需要。J2EE Web 核心技术系列教材分别涉及 XHTML 与 XML 应用开发、Web 组件与框架开发技术等方面的内容，这些技术课程都是目前高校计算机学院及软件学院二年级和高职三年级的通用课程。

作者也会对 J2EE Web 核心技术系列教材进一步扩展，编写涉及 Java 2 语言及面向对象编程应用、J2SE 实用开发技术等方面的教材，这些编程语言、应用技术课程都是目前高校计算机学院及软件学院一年级和高职二年级的通用专业基础课程。为高校师生提供多层次的教学支持和技术帮助，提升高校计算机学院和软件学院的教学质量。

（2）目前高校使用的 Java 类的教材内容及技术都比较陈旧

J2EE 技术规范从 1997 年发布至今已经有 13 年，Java 与 J2EE 技术规范本身也在不断地进行完善和升级，已经发生了根本性的改变。但目前许多高校在 Java 与 J2EE 相关技术与应用的教学中所采用的教材太“语法化和原理化”或者直接采用技术参考资料兼作教材，而且还缺少软件工程中所倡导的“规范性”的内容，如“流程和规范”、“思想和原则”、“技术和应用”以及“效率和质量”、“协同和协作”等方面。

作者本人特向清华大学出版社提出编写“J2EE Web 核心技术”课程教学系列教材的计划，该计划也是对作者的“软件工程专业项目实训”系列教材（已在 2008 年由清华大学出版社出版）、“软件工程专业课程设计”系列教材（已在 2009 年由清华大学出版社出版）的进一步丰富。该系列教材的出版将为学生进一步学习其他软件开发专业课程和今后从事软件开发工作打下坚实的基础，提升学生的职业技能，提高高校学生的就业竞争力。

2. 本系列教材在内容方面的主要特色

（1）系列教材所涉及的技术主题定位

J2EE Web 核心技术系列教材在技术主题的定位方面，继续沿用已经出版的“J2EE 项目实

训”、“J2EE课程设计”系列教材的技术风格，选择目前比较热门的Web 2.0技术、主流的J2EE平台中的各种核心技术，并结合项目开发具体实例进行详细和深入介绍。

另外，为了使得本系列教材能够适应不同层次的读者群的要求，每个案例都是对某类问题的解决方法的模板。

（2）与同类技术参考书有本质的不同

目前高校J2EE平台软件开发类教材很少，学校采用的几乎都是市场上的“店销”科技书（技术参考书）。但科技书往往只追求技术内容的前沿性，而缺少完整的知识体系，也没有课后练习、教学指导和学习参考，不适合课堂的日常教学。本系列教材不仅在内容的选择方面有别于一般的“店销”技术参考书，而且还为教师和学生提供日常教学和学习指导——每章都附有教学重点、学习难点、教学要点和学习要点等内容，另外，每章的案例都提供有程序源代码，能够更好地帮助授课教师进行日常教学，提高教学水平和教学效果。

（3）系列教材中文字表达的特色

J2EE Web核心技术系列教材在内容的组织和案例的选择方面，力求避免抽象的理论介绍，而是以目前企业级的软件项目开发实现过程中所涉及的J2EE各个核心技术方面的知识为基本素材展开讲解；考虑到高校低年级学生的知识水平和理解力，在教材的文字表达方面采用图文并茂的写作风格。这样能够使学生真正掌握和了解目前企业级的应用系统开发中所需要的知识和技术，授课教师不仅了解教什么，也知道应该如何教。

3. 本系列教材在写作风格方面的主要特色

J2EE课程是一门重要的计算机专业及软件工程专业的专业课、专业限选课程，几乎所有高校的计算机学院计算机应用专业、软件学院软件工程专业都开设了此方面的课程。作者结合自身多年的一线教学活动实践和对多所高校软件学院的本科生和研究生的教学指导，为高校师生提供一套价格适中、内容全面和系统的J2EE开发类的教材。

本系列教材在内容的选择方面不但包括J2EE核心技术，还包括目前在软件企业中广泛应用的J2EE框架、开源开发工具等方面的内容。书中案例丰富，充分体现了现代软件工程教育中的

CDIO理念：构思（Conceive）、设计（Design）、实现（Implement）和运作（Operate）。

为了能够在有限的篇幅里讲述最多的技术内容，本教材的写作秉承课程讲授风格，重点突出、内容精练、案例丰富，对案例的实现都附有详细的实现过程的屏幕截图；作者在多年的J2EE一线教学过程中，根据学生的课后反馈对课程讲义内容不断地进行调整、改进、完善，此系列教材的编写以实际授课的课程讲义为基础。内容的安排不仅适合学生的学习和课后实践，也符合学生的学习习惯和知识水平。

教材中收录的各章练习题难易适中，工作量也适中，有利于学生在课后巩固所学的课堂知识。

4. 关于本书的内容介绍

本书共9章，内容分为3大部分。前4章主要涉及应用“XHTML＋CSS＋Div＋JS”构建Web网站必须掌握的HTML/XHTML、CSS、Div和JavaScript语言等方面的技术及应用；第5、6章的内容主要涉及XML技术及文档类型定义(Document Type Definition，DTD)、XML Schema技术及应用等XML语法相关内容；第7、8和9章的内容与XML的转换和解析技术有关，主要包括XML XSLT技术及应用、XML XPath技术及应用、XML DOM技术及应用等。

对于Java平台中的XML解析实现技术(如SAX和JDom等)方面的内容，由于作者已经在《J2EE课程设计——技术应用指导》一书(见本书的参考文献)第6章“XML解析技术及在项目开发中的应用”中有详细介绍，因此在本书中不再重复介绍。

5. 适宜的读者对象

本系列教材适合作为承担国家技能型紧缺人才培养培训工程的高等职业院校和示范性软件学院的计算机应用与软件工程专业的J2EE技术平台下的应用开发类课程的教学和学习教材，也可作为自学和急需了解J2EE技术平台的软件项目开发和实现的相关技术和知识的技术人员的参考书。当然，本书也适用于各类职业技能培训机构的J2EE应用开发类培训课程的教材。

6. 本书的阅读方法

由于本书以及本系列教材侧重于“技术应用及开发实现”，在教材中将会出现大量的案例和教学示例。因此，建议读者在阅读本书时最好能够按照书中所给出的各个示例中的设计方法和实现步骤进行实际练习，并完成各个章节中提供的练习，这样的学习效果会比较好。

本书中所附的各个截图都出自相关的截图软件，作者未作任何的改动和拼接。

7. 致谢

在J2EE Web核心技术系列教材的编写过程中，得到了清华大学出版社有关编辑的大力帮助和支持，他们对本系列教材的选题、内容及编写风格等都提出了许多宝贵的建议，在此一并感谢。同时，本系列教材的出版也得到了中科院计算所职业培训中心王健华校长的大力支持，感谢王校长给作者在工作上的帮助和指导，也感谢培训中心的各位同事和教师的大量支持。

中科院计算所职业培训中心长期从事校企合作人才培养、企业内训、职业技能提升、项目管理等领域的咨询、教学和技术服务方面的工作，也会不断地为高校提供实用型、高质量的教学辅导参考教材。

编　者

2010 年 2 月

CONTENTS

目录

CONTENTS

C O N T E N T S

CONTENTS

CONTENTS

第1章　HTML、XHTML和DHTML标签语言

CHAPTER 1

W3C颁布的HTML标签元素按功能划分基本上可以分为三类：用于表示HTML文档结构的标签元素，如table、li、div和frame等；用于表示HTML文档显示样式的标签元素，如font、strong、style等；用来创建用户界面实现人机交互功能的标签元素，如form、a、script等。而XHTML是更严谨、更规范化的HTML规范和标准，提出XHTML的主要目的是希望Web开发人员能够设计并实现出比较“严谨”和“规范”的HTML页面。

DHTML不是由W3C提出的技术标准，而是由浏览器厂商根据自己的需要在各自的浏览器中支持的整合技术，它是目前比较成熟的Web客户端应用技术。应用DHTML技术，可以实现在浏览器客户端直接动态地更新Web网页中的信息内容、排版样式、播放动画等，而无须经过服务器端程序的处理。

HTML、XHTML、DHTML和Web 2.0等术语的真正内涵到底是什么？Web开发者如何快速地进行Web页面开发？Web 2.0时代有哪些主要的代表技术？对这些问题的解答构成本章的重点内容，当然，还包括如何更好地教授本章的内容和如何有效地学习本章的内容。

1.1　HTML标签语言及典型标签的应用

1.1.1　超文本标签语言及规范

1. 超文本标签语言

超文本标签语言（Hyper text Markup Language，HTML）起源于标准通用的标签语言（Standard Generic Markup Language，SGML），它是W3C发布的一种标签（或称为标记）形式的语言技术标准，提出该标准的主要目的是希望能够实现在各种不同形式的网络环境之间、不同文本格式文件之间相互进行数据交换。Web开发人员可以在页面中应用各种HTML标签元素标识和说明各种形式的文字、图形、动画、声音、表格、超链接等类型的信息，最终达到对这些信息内容的显示与具体的显示实现设备的无关性。

HTML 目前的版本为 HTML 5，并允许在 HTML 页面中直接嵌入操作音频、视频、图片的函数脚本程序，并进一步改进了互操作性和减少开发成本。HTML 标签语言规范中的各个标签通常是英文单词的全称（如描述表格的 table 标签等）或缩略语（如 b 代表粗体 Bold 等），并且要放在单书名号（“＜”）中。如下示例中的＜table＞标签描述了一个宽度为整屏宽度的表格。

```
<table width="100%" ></table>
```

一个 HTML 文档文件是由一系列的 HTML 标签所构成的，并且标签名不区分字符的大小写，而且也允许带有不同名称的属性和为各个属性提供不同的值。

2. 格式良好的 HTML 文档的基本组成

（1）成对的＜html＞标签元素为 HTML 页面中的顶级根标签

任何一个格式良好的 HTML 文档都必须要以＜html＞标签开始并最终以＜/html＞标签结束（这也就是 HTML 标签的成对特性），其中的＜html＞标签表示该页面文档是以超文本标签语言编写而成的。尽管目前有许多 Web 浏览器都可以自动地识别 HTML 文档，并不严格要求在页面中必须要有＜html＞标签，也不对该标签进行任何实质性的语法检查操作，但是为了使得所开发出的 HTML 文档能够规范化和适应各种不同类型的 Web 浏览器，作者还是建议读者遵守该设计规则。

（2）成对的＜head＞标签标识 Web 页面的头部信息

在成对的＜html＞标签之间的内容又可以被进一步划分为两个不同的部分：页面头部（Head）和页面内容主体（Body）。由成对的＜head＞和＜/head＞标签所标识的信息为页面头部信息，其中的各个标签的主要作用是描述本页面的特征和为浏览器正确解析本页面而提供的描述性信息。头部信息不在浏览器的窗口内显示输出。

（3）成对的＜body＞标签标识页面实际显示输出的信息内容

由成对的＜body＞和＜/body＞标签所标识的内容为页面主体内容，其中的各个标签元素标识在浏览器窗口区域中要显示输出的各种内容信息，它们可以为文字、超链接、表格、图像等多种不同的形式，并且还可以格式化这些信息内容。

下面分别介绍非框架集和框架集两种类型的页面文档的基本结构及基本的语法要求。

3. 非框架集 HTML 文档的结构及基本的语法要求

（1）非框架集 HTML 文档的基本结构

在成对的＜html＞＜/html＞标签中必须要包含成对的＜head＞＜/head＞和＜body＞＜/body＞标签，其中成对的＜head＞＜/head＞标签是该 HTML 文档的头部标签集，在浏览器窗口中，头部标签所描述的信息在浏览器窗口的正文区域中是不被直接显示输出的。当然，在此标签集中还可以插入其他标签，如页面标题＜title＞标签，用以说明本页面文件在浏览器窗口中显示时的标题栏的信息。因此，一个 HTML 文档的基本结构如下：

＜html＞ ——HTML 文档开始

<head> ——页面头部信息开始
页面头部信息(给浏览器提供相关信息)
</head> ——页面头部信息结束
<body> ——页面主体内容开始
页面主体内容(在浏览器窗口中显示输出)
</body> ——页面主体内容结束
</html> ——HTML 文档结束

(2) 体现非框架集页面文档的结构及基本的语法要求的示例

例 1-1 中的 HTML 文档结构的标签代码示例基本上反映了非框架集页面文档的结构及基本的语法要求,在页面内容区中设计一个表格,并在单元格中包含超链接。

例 1-1　一个体现非框架集 HTML 文档结构的标签代码示例

```
<html><head> <title>这是我的 HelloWorld 页面 </title> </head>
    <body><table>
            <tr><td><a href = "http://www.sina.com" >新闻 </a></td>
                <td><a href = "http://www.cctv.com" >电视 </a></td>
                <td><a href = "http://www.tuduo.com" >电影 </a></td>
            </tr></table>
    </body>
</html>
```

例 1-1 中成对的<body></body>标签之间的标签及文本即为浏览器窗口正文区域中的信息及格式定义,并且在浏览器的页面内容区域中被显示输出。在本示例中包含若干个超链接标签,产生出 Web 应用系统中常见的导航菜单栏效果。例 1-1 示例页面文档在浏览器中执行后的结果如图 1.1 所示。

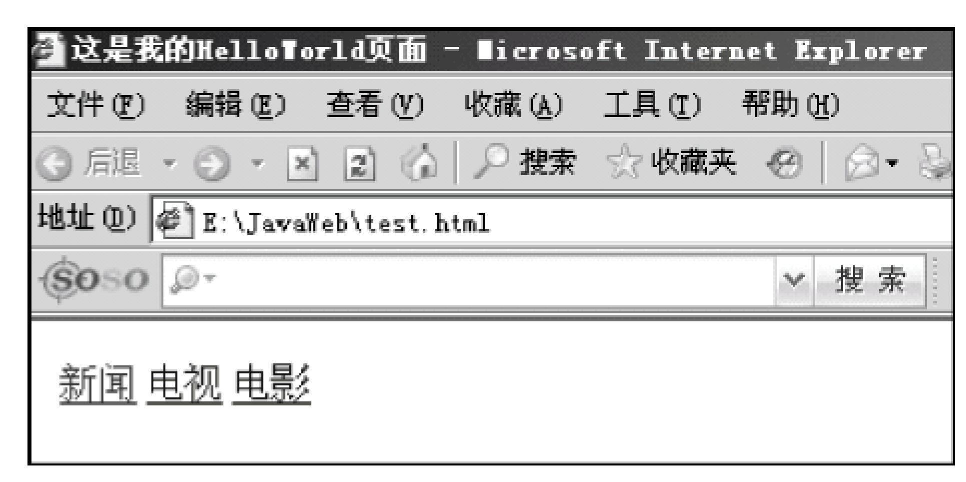


图 1.1　例 1-1 示例页面文档在浏览器中执行后的结果

4. 框架集 HTML 文档的结构及基本的语法要求

(1) 框架集 HTML 文档的基本结构

在 Web 应用系统的页面开发中,为了能够实现将浏览器窗口在水平和垂直方向上拆分为多个不同的独立显示区域,可以应用 HTML 中的框架集技术设计和实现该页面,因为一个框架集 HTML 文档可以在浏览器窗口中同时显示多个不同的子窗口页面文件。

HTML 标签集中的<frameset>标签定义了框架集中的各个子窗口的结构和布局,而其中的<frame>标签则具体定义出框架集中的某个特定子窗口所对应的目标 HTML 文件,该页面同样也要满足在例 1-1 中所示的 HTML 文档的结构和基本的语法要求,并且所有的<frame>标签必须要被包含在成对的<frameset>标签内。

(2) 体现框架集 HTML 文档结构及基本的语法要求的示例

例 1-2 中的框架集示例页面标签中的外层<frameset>标签实现把整个浏览器的页面

窗口在水平方向上分成左右两部分，其中左面子窗口的宽度为 80 像素，而右面子窗口的宽度为浏览器窗口中剩余的宽度量；同时又将右面窗口在垂直方向上拆分为上下两个子窗口，其中顶端子窗口的高度为 80 像素，而下面子窗口的高度为浏览器窗口剩余的高度量。

例 1-2　一个体现框架集 HTML 文档结构的示例

```
<html><head><title>框架集 HTML 文档的基本结构要求</title></head>
<frameset rows = " * " cols = "80, * " frameborder = "yes" border = "1" framespacing = "0">
   <frame src = "left.html" scrolling = "no" noresize = "noresize" id = "leftFrame"/>
   <frameset rows = "80, * " frameborder = "yes" border = "1" framespacing = "0">
      <frame src = "top.html" scrolling = "no" noresize = "noresize" id = "topFrame" />
      <frame src = "main.html" id = "mainFrame"   />
   </frameset>
</frameset></html>
```

由于在<frameset>标签中不显示任何有效的文字信息，因此不必将<frameset>标签放入到<body>标签中。<frameset>标签中的 rows 属性决定行的数目(水平分割)，而 cols 属性决定列的数目(垂直分割)，frameborder 属性设置是否需要边框线，border 属性设置边框线的粗细，framespacing 属性设置各个子窗框之间需要保留的空白距离。例 1-2 所示的框架集 HTML 在浏览器中执行后的预览效果如图 1.2 所示。

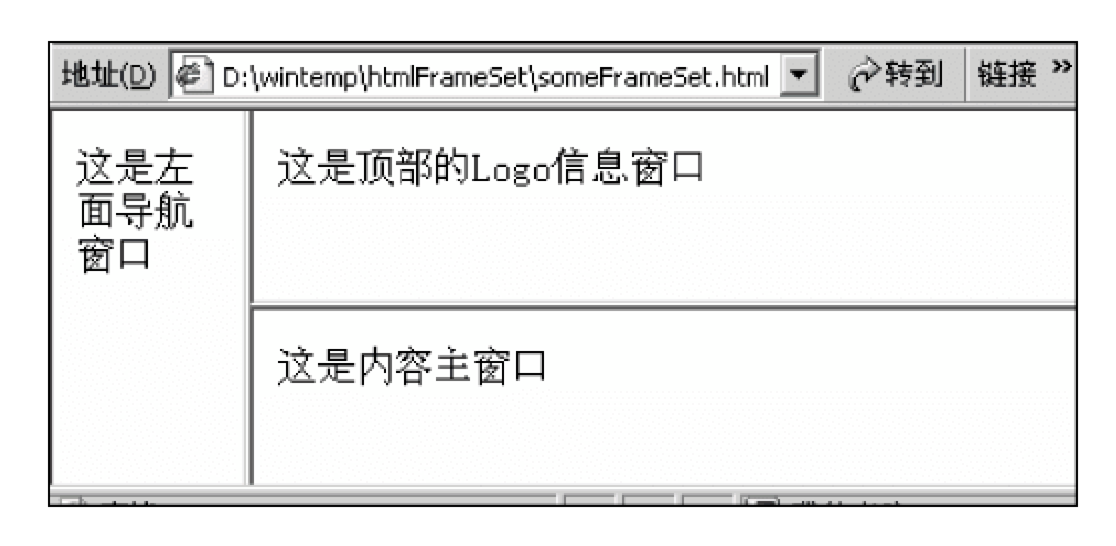


图 1.2　例 1-2 示例 HTML 文档在浏览器中执行后的结果

(3) 利用框架集<frameset>标签可以分隔不同类型的信息

例 1-2 的框架集文档中所指示的每一个子窗口都由一个单独的 HTML 文档文件所代表，本示例为 left.html(标识左窗口)、top.html(标识右边顶端窗口)和 main.html(标识右边下端窗口)文件。读者可以根据在 Web 项目中对输入界面的设计要求开发并实现各个子窗口 HTML 页面文件，并且可以在每个子窗口中显示不同类型的内容信息。

例 1-2 示例页面中的左面框架代表某个系统的全局导航信息的页面内容，在浏览的整个过程中将保持固定不变，访问者在左侧树型导航菜单栏中单击某个超链接后，将在右下侧的内容框架子窗口中显示出当前页面信息的文档内容(作者在此没有附有各个子窗口的页面内容)。

当然，如果需要在一个页面文档中嵌入另一个 HTML 文档中的内容，但又不希望产生出窗口被明显分隔的显示效果，可以应用<iframe>标签。<iframe>标签的主要作用是在当前页面文档区域中标签出一个独立的显示区域，显示其他页面文档中的内容，产生出类似电视节目中的"画中画"的显示效果，但<iframe>标签是非 W3C 标准的标签。

1.1.2 Web应用系统中的典型HTML标签及应用

1. 超链接标签<a>及应用

(1) Web页面中超链接(HyperLink)的主要作用

Web页面中的超链接能够实现将各个网页文件互相链接，用户单击被称为超链接的文本或图像的内容就可以浏览目标页面文件中的内容信息。当然，超链接除了可链接文本和图像外，也可链接其他形式的多媒体信息，如声音、视频、动画等。

(2) 超链接<a>标签及属性的定义语法

在HTML标签中采用<a>标签表示超链接，并且采用成对的<a>和标签建立超链接。超链接<a>标签的语法格式及主要的属性项目如下：

```
<a href = "目标资源的URL地址"  target = "目标窗口名称"  title = "tip动态显示功能"> 超链接显示的内容</a>
```

其中的href属性定义了这个超链接实际所指向的目标资源的URL(Uniform Resource Locator，统一资源定位符)地址，而target属性用于指定打开该链接的目标资源的窗口方式(默认方式是在浏览器的当前窗口内显示)。下面为target属性的可能取值及对应的功能说明。

- _parent：在上一级窗口中打开，一般应用在框架集的框架页中。
- _blank：在一个新的空窗口中打开。
- _self：在同一个帧或当前窗口中打开(默认为该方式)。
- _top：在浏览器的顶级窗口中打开。

“超链接显示的内容”可以是文本或者图像，并且在浏览器中对文本的默认显示方式为带下画线的文本且与页面中其他正常的信息文字的颜色不同；而图像超链接的默认显示风格为带有边框线，当然也可以通过设置样式风格屏蔽边框线。如果需要用图像做超链接，只需要把定义图像的<img>标签嵌套在之间就能实现图像超链接的效果。

当用户的鼠标指向“超链接显示的内容”位置处时，鼠标就会自动地变成手状；用户如果单击“超链接显示的内容”元素，则可以访问指定的目标资源文件，浏览器也会相应地加载该目标资源文件。

(3) 利用超链接<a>标签定义一个命名锚实现页面内超链接

超链接<a>标签不仅可以实现从一个页面链接到另一个独立的页面的跳转，也可以实现在同一个页面内的不同内容所在的区域之间链接跳转。此时，只需要在页面某处定义一个命名的锚，然后再定义一个到该命名锚的正常超链接即可。下面给出了实现此功能要求的HTML代码片段示例。

```
<a id = "oneNamedAnchor">命名锚所指示的文本内容</a>
<a href = "#oneNamedAnchor">跳转到锚标签为oneNamedAnchor的指定位置</a>
```

（4）超链接<a>标签的典型应用示例

- 带动态帮助提示信息（Tooltip），利用该信息可以为超链接提供说明性的文字，当操作者的鼠标悬停在该超链接上时，浏览器将自动显示出该帮助提示信息：

```
<a href = "oneImage.jpeg" title = "单击可以查看详细信息">详细的放大图</a>
```

- 超链接到某个 E-mail 地址：

```
<a href = "mailto:abc@abc.com">联系我们</a>
```

- 将某个图片作为超链接：

```
<a href = "#"><img src = "someTwoImage.jpeg"></a>
```

2. 图像标签<img>和背景图像的应用

（1）Web 浏览器目前所支持的各种图像文件格式

Web 浏览器可以显示的图像文件的格式主要有 JPEG（Joint Photographic Experts Group，联合图像专家组）、BMP（英文 Bitmap（位图）的简写，同时也是 windows 环境中交换与图形有关的数据的一种标准）、GIF（Graphics Interchange Format，图像互换格式）等形式，但由于 BMP 格式的图像文件存储的容量比较大、传输慢，一般在 Web 页面中不应用它；而 JPEG 图像由于支持数百万种颜色，即使在传输过程中丢失部分颜色数据，也不会在图像显示质量上有明显的差别，但图像文件的容量比 GIF 格式的图像文件要大；GIF 图像由于仅包括 265 种色彩，尽管在显示质量上没有 JPEG 格式的图像高，但图像文件的容量比较小，下载速度也比较快，并且还支持动画效果及背景色透明等技术特性，因此也同样被广泛地应用在 Web 页面中。

（2）利用<img>标签实现在网页中插入独立的图像

当浏览器读取到页面文档中标识图像的标签时，就会显示由其中的 src 属性所指定的目标图像文件而产生出“图文混排”的页面排版效果。<img>标签主要有如下的属性项目。

- src：图像文件的 URL 地址。
- alt：鼠标指向图像时的动态提示文字。
- width：图像的宽度，一般设为图像的真实大小，以免失真。
- height：图像的高度，同样也应该设为图像的真实大小，以免失真。

Web 开发人员如果没有设置<img>标签中的 width 或者 height 属性值，浏览器将会以图像的真实大小显示输出该图像。

（3）页面中某个区域的背景图像

在 Web 网页设计中，开发人员除了可以利用颜色做背景外（但不能产生出渐变的颜色效果），也可以用图像设置背景（该方法不仅可以产生出渐变的颜色效果，而且还可以产生出背景图）。设置背景图像的方法是将某个标签的 background 属性指定为目标图像文件的 URL 地址，如<div background="图像文件的 URL 地址">，其中的“图像文件的 URL 地址”指定图像文件的文件目录及文件名等定位信息。

在如图 1.3 所示的某个客户关系管理系统中的页头区域设计中，Web 开发人员不仅利用<img>标签显示出企业的徽标(其中的太极图)，而且也使用<div>标签设置出“蓝天白云”的背景图效果。



图 1.3　网页中独立的图像及背景图示例

3. 利用<marquee>标签产生滚动的文字

(1) 与表格有关的各标签的语法

在 Web 页面设计中，为了能够在有限的显示区域中显示更多的文字信息，可以利用<marquee>标签制作出具有滚动效果的文字，类似于电视节目中滚动字幕的效果。<marquee>标签默认的滚动方向是从浏览器窗口的右侧向左侧方向滚动，但如果要改变滚动文字的滚动方向，则可以设置<marquee>标签中的 direction 属性的值。<marquee>标签中的主要属性项目及可能的取值的含义如下。

- direction：设定滚动文字的滚动方向，可以为“上”、“下”、“左”、“右”4 种方向。示例：<marquee direction="left">从右向左滚动的文字</marquee>。
- align：设定滚动文字的对齐方式，可以为“居左”、“居中”、“居右”、“靠上”和“靠下”几种形式，如 align ="left"表示“居左”。
- behavior：设定滚动文字的滚动方式，可以为 behavior="scroll"，表示由一端滚动到另一端；behavior="slide"表示由一端快速滑动到另一端且不再重复；behavior="alternate"则表示在两端之间来回滚动(摆动方式)。
- height 和 width：设置滚动区域的范围，其中 height 属性设定滚动区域的高度，而 width 属性设定滚动区域的宽度。
- scrollamount：用于设定滚动文字的滚动速度，也就是滚动文字每次移动的长度，单位是像素。
- scrolldelay：用于设定两次滚动之间的延迟时间(以毫秒为单位)，这个延迟时间设置得越小，滚动的速度也就越快。
- loop：用于设定滚动的重复次数，当 loop=－1 时表示一直滚动下去(周而复始地滚动)，直到页面更新为止。

(2) 利用<marquee>标签产生滚动文字的代码片段示例

下面的代码片段示例将某个客户关系管理系统中的“系统最新公告信息”设置为滚动效果，并且带有超链接，采用浏览器中默认的滚动方向和默认的流动速度。

```
<marquee onmouseover = "this.stop();" onmouseout = "this.start();">
<a href = "#">系统最新公告：蓝梦集团成功香港上市，为进军国际化市场迈出了可喜的第一步！
单击可以浏览详细的信息……</a></marquee>
```

当用户用鼠标指向该滚动超链接的文字时，它将自动地停止流动；而当鼠标离开该滚动

超链接的文字时，它又自动地进行滚动，这是由于在本示例中添加了鼠标响应事件代码。本示例在浏览器中浏览时的效果如图 1.4 所示，并且是作者用鼠标指向后暂停时的图示效果。

图 1.4　某个客户关系管理系统中滚动“系统最新公告”的预览效果局部图示

4. 表格<table>标签及应用

（1）普通的表格是通过<table>、<th>、<tr>和<td>等标签来实现的

<table>标签用于定义表格的开始和结束（整体结构和特性）；而<th>标签用于定义表格头部行中的单元格，并且<th>标签必须放在<tr>标签内；<tr>标签用于定义表格的行，并且在其中需要包含由若干个<td>标签所定义的表格单元格，<td>标签也必须放在<tr>标签内。下面列出与表格<table>标签相关的各个主要的属性项目及对应的功能含义。

- width/height：指定表格或某一单元格的宽度/高度。
- border：指定边框线条的宽度。
- bordercolor：指定边框线条的颜色。
- bgcolor：指定表格或某一单元格的背景颜色。
- background：指定表格或某一单元格的背景图片。
- align：设置单元格对齐方式。
- cellspacing：设置单元格间距。
- cellpadding：设置单元格内容与单元格边界之间的距离。
- colspan/rowspan：实现跨列/跨行单元格的显示风格，设置跨列/跨行的数目。

在 Web 应用系统的页面设计和实现中，表格<table>标签不仅可以实现将数据以行/列规整的形式显示输出，而且还能够进行页面整体的布局定位。下面为一个利用 HTML 表格<table>标签实现以表格行/列规整的方式显示系统中查询结果数据的标签代码片段示例（只列出了表格中的表头部分的标签，省略了表体部分的数据及标签）。

```
<table width="100%" border="1" ><tr>
        <th width="12%">账号</th>        <th width="8%">姓名</th>
        <th width="12%">开户时间</th>    <th width="13%">存期(月)</th>
        <th width="19%">身份证 ID</th>   <th width="10%">账户余额(元)</th>
        <th width="10%">状态</td>        <th width="16%">系统注册 ID</th>
  </tr></table>
```

（2）创建跨多行、多列的表格单元格

如果在<th>或<td>标签中加入 rowspan 或 colspan 属性项目，可以创建出跨表格行或跨表格列显示效果的表格。比如 colspan="2"表示表格中的某个单元格跨越表格两个列的宽度；而 rowspan="2"则表示这一单元格跨越表格两个行的高度。

（3）创建行分组的表格

表格按行分组主要是由表头<thead>标签、表格主体<tbody>标签和尾注<tfoot>标签三个部分组成的。<thead>、<tbody>标签的属性和<th>、<td>标签的属性是一样的。下面为一个行分组的表格标签代码片段示例，在浏览器中的显示结果如图 1.5(a)所示。

```
<table border = "1">
    <thead><tr><th>姓名</th><th>年龄</th><th>籍贯</th><th>毕业学校</th></tr>
    </thead>
    <tbody><tr><td>张三</td><td> 23 </td><td>北京</td><td>蓝梦大学</td></tr>
    </tbody>
</table>
```

（4）创建列分组的表格

通过应用<colgroup>标签实现将表格按列方向进行逻辑分组和组合，每组可包含一列或几列单元格，然后可以对各组分别设置显示格式，以便能够统一格式化定义这些列，从而减少对样式格式重复设定的次数。

<colgroup>标签有两个主要的属性：一个为 span 属性，表示该列分组中具体的列数，默认值为 1；另一个为 align 属性，指定该列分组中所有的单元格内容的对齐形式，可能的值为“left”(左对齐)、“center”(水平居中)和“right”(右对齐)之一。

下面的代码片段示例采用两个<colgroup>标签把一个表格按列方向分为两组，其中第二组包含两列单元格，为每个列分组都设置了不同的样式风格定义(背景为红色和白色)，注意其中黑体所标识的标签。在浏览器中的显示结果如图 1.5(b)所示。

```
<table border = "1">
     <colgroup align = "left" style = "background - color: # FF0000"></colgroup>
     <colgroup span = "2" style = "background - color: # ffffff"></colgroup>
     <tr><th>姓名</th><th>年龄</th><th>学校</th></tr>
     <tr><td>张三</td><td> 21 </td><td>蓝梦大学</td></tr>
</table>
```

（5）定义表格的标题

利用<caption>标签可以设置表格的标题，并且标题的定位由 align 和 valign 属性项目决定，默认的定位属性取值都为“center”(水平居中)。其中 align 属性设置表格标题文字水平方向的对齐形式：取值为 left，将产生出“左对齐”的显示效果；取值为 center，将产生出“水平居中”的显示效果；取值为 right，将产生出“右对齐”的显示效果。而 valign 属性则设置表格标题文字垂直方向的对齐形式：取值为 top，将产生出“顶部对齐”的显示效果；取值为 bottom，将产生出“底部对齐”的显示效果。

姓 名	年龄	籍贯	毕业学校
张三	23	北京	蓝梦大学

(a) 表格行分组示例图

姓名	年龄	学校
张三	21	蓝梦大学

(b) 表格列分组示例图

员工基本信息表

姓 名	年龄	籍贯	毕业学校
张三	23	北京	蓝梦大学

(c) 带标题的表格示例图

图 1.5　表格示例图

<caption>标签必须位于<table>标签之内，并且在表格行<tr>标签之前。下面为应用<caption>标签为某个数据表格定义标题文字的代码片段示例，注意其中黑体所标识的标签。在浏览器中的显示结果如图 1.5(c)所示。

```
<table border = "1">
    <caption align = "center" valign = "top">员工基本信息表</caption>
    <thead><tr><th>姓名</th><th>年龄</th><th>籍贯</th><th>毕业学校</th></tr>
    </thead>
    <tbody><tr><td>张三</td><td> 23 </td><td>北京</td><td>蓝梦大学</td></tr>
    </tbody>
</table>
```

(6) 利用表格实现页面内容的布局定位

表格<table>标签不仅可以实现以行、列的方式规范数据的显示，也可以利用表格实现页面内容的布局定位，并且利用表格可以相互嵌套的特性实现复杂的页面布局定位。

利用表格实现复杂的页面内容布局定位的基本思路主要是：由外部总表格规划整个页面(或者某个区域)的整体结构和布局，而由其中嵌套的子表格负责各个子区域的局部布局，并根据区域划分的情况逐步细化。当然，此时不应该显示出表格的边框线，也就是要将<table>标签中的 border 属性取值为 0。

例 1-1 其实就是一个利用表格<table>标签为某个系统中的导航菜单栏实现布局定位的部分代码片段示例，只是作者对该示例进行了简化。但由于表格是将“内容”(如例 1-1 示例中的“新闻”、“电视”和“电影”等)和“表现”(如例 1-1 示例中的<table>、<tr>和<td>等标签)相互混合在整个<table>标签内，这不利于将“页面内容”的表现实现多样化显示。

因此，现在的 Web 页面设计技术基本上都抛弃了利用表格实现页面内容的布局定位技术实现的方法，而是采用“CSS+Div”布局定位方式。其中利用<div>标签定义要显示的内容，而借助 CSS(Cascading Style Sheets，层叠样式表)实现内容的显示风格控制，最终达到“页面内容”和“页面表现风格”相互分离的设计效果。

5. 利用<div>和<span>标签实现页面内容的布局定位

(1) <div>标签用于定义和划分页面中的“页面区域”

<div>标签是用来为 HTML 文档内大块(区域)的内容提供结构和背景描述的标签，它可以把整个页面文档分割为若干独立的、不同的显示区域，并且通过应用层叠样式表定义这个区域内信息内容的显示风格。

(2) 采用“CSS+Div”页面布局定位技术构建 Web 网站系统页面

利用<div>标签构建好整个 Web 网站系统中各个版块页面的布局之后，再用 CSS 给它添加样式风格定义，最终达到将页面中的各个“信息内容”(由<div>标签标识)和“表现风格”(由 CSS 文件中的各个属性项目规定)相互分离，类似于软件系统开发中模型—视图—控制器设计模式(Mode View Control，MVC)中的“M(模型)”和“V(视图)”相互分离的设计效果。如下为对图 1.1 中的页面导航菜单栏应用<div>标签实现相同的布局定位效果的部分代码片段示例。

```
<div id = "navMenuBar"><ul><li><a href = "http://www.sina.com">新闻</a></li>
```

```
                    <li><a href="http://www.cctv.com">电视</a></li>
                    <li><a href="http://www.tuduo.com">电影</a></li>
</ul></div>
```

如果希望将上面的代码片段在浏览器中显示，最终能够达到与图 1.1 所示的显示效果相同时，还需要应用 CSS 定义它们的显示风格，作者在此没有给出对应的 CSS 定义文件的属性项目。

(3) <div>和<span>标签在应用方面的主要差别

<div>和<span>标签都可以实现页面内容的布局定位，但它们在应用方面有着本质差别：<div>标签是块级别标签，由<div>标签所标识的每个内容独占一行的显示区域；而<span>标签则是行内标签，由<span>标签所标识的每个内容并不独占一行的区域，而是顺序地排列在一行中。

如下为对图 1.1 中导航菜单栏中的各个超链接应用<span>标签实现行内布局定位的部分代码片段示例，添加对应的 CSS 样式定义后同样能够达到与图 1.1 相同的显示效果。

```
<div id="navMenuBar"> <span><a href="http://www.sina.com">新闻</a></span>
                      <span><a href="http://www.cctv.com">电视</a></span>
                      <span><a href="http://www.tuduo.com">电影</a></span>
</div>
```

在上面的<div>标签中只有对“内容”(本示例为超链接)的描述标签，而没有对这些“内容”最终如何“显示”的定义和描述标签，因此“内容”和“显示”相互分离。

6. 利用<ul>标签定义符号列表、<ol>标签定义编号列表

(1) 标签和<li>标签组合用于定义符号列表

这种形式的列表中每个列表项前面可以是一个圆点、方块等符号，其中<ul>标签中的 type 属性项目决定列表符号的类型，可能的取值为 disk(圆点列表符)、circle(圆圈列表符)和 square(方点列表符)。例 1-3(a)所示为一个利用<ul>标签和<li>标签组合定义符号列表的代码片段示例。

(2) 标签和<li>标签组合用于定义编号列表

这种形式的列表中每个列表项前面设置编号的具体形式，可以是数字编号，也可以是字母编号。其中<ol>标签中的 type 属性决定编号的类型，可能的取值可以为 1(表示采用数字编号)、I(表示采用大写罗马数字编号)、i(表示采用小写罗马数字编号)、A(表示采用大写字母编号)和 a(表示采用小写字母编号)。例 1-3(b)所示为一个利用<ol>标签和<li>标签组合定义编号列表的代码片段示例。

例 1-3　标签，<ol>标签应用举例

(a) 符号列表代码片段示例

```
<ul type="circle">
  <li>J2ME</li>
  <li>J2SE</li>
  <li>J2EE</li>
```

```
  <li>J2EE 框架</li>
</ul>
```

（b）编号列表代码片段示例

```
<ol type="I">
  <li>J2ME</li>
  <li>J2SE</li>
  <li>J2EE</li>
  <li>J2EE 框架</li>
</ol>
```

7. Web 表单<form>标签的基本语法及应用

（1）Web 表单的作用

Web 表单是 B/S(Browser/Server,Web 浏览器/Web 服务器)体系架构的企业应用系统中用户与 Web 系统之间进行人机交互的主要图形用户界面(Graphical User Interface,GUI)元素,应用系统通过 Web 表单可以收集用户的各个请求参数,并对客户端的请求加以对应的后台处理。

用户提交某个表单后,浏览器将自动收集该表单中的各个信息并发送到 Web 服务器端相关的请求响应处理程序,服务器端相关的应用程序(比如采用 CGI、PHP、ASP 或 JSP 等技术实现的程序)会进行对应的功能处理,将请求的信息保存到数据库表中或者将处理后的结果信息再返送回客户端的浏览器中显示输出,用户也就看到了这次请求的处理结果信息,这也就是常见的 Web 应用系统中的人机交互的基本过程。

（2）<form>标签中的各个主要属性

<form>标签在浏览器中显示时也是一个区块,并且也独占一行显示区域(除非嵌套到另一个大的区块内)。下面为利用<form>标签定义的某个表单的代码片段示例。

```
<form action="/webcrm/sendBBSInfo.jsp" method="post"
          id="sendBBSInfoForm" onSubmit="checkAllItemInform(this);">
    表单内各个控件标签的定义,在此省略
</form>
```

其中的 action 属性定义了该表单请求的 Web 服务器端程序的 URL 地址,本示例为某个 JSP 页面文件;method 属性指定了本表单的请求方式,可以为 get 请求和 post 请求;而 id 属性定义了该表单的名称,有助于在 JavaScript 脚本程序中对表单进行动态编程控制;onSubmit 属性为该表单添加一个事件响应的 JavaScript 脚本函数,当该表单提交后将首先执行由 onSubmit 属性所指定的 JavaScript 脚本程序代码。

8. 与 Web 表单相关的各个控件标签

由于<form>标签只是一个表单的区块定义,可以将它理解为一个应用程序的窗口区域;而操作界面中的具体输入控件则由包含在<form>标签内的各个控件标签定义。

（1）<form>标签内的<input>标签

<input>标签产生各种“输入”类的 Web 表单控件,用于搜集用户请求的各种输入信

息。而不同的控件类型则是根据它的 type 属性的取值进行区分的，主要分为单行文本输入控件、密码输入控件、单选按钮、复选框、隐藏输入控件、普通按钮、提交按钮、取消按钮、图片提交按钮、文件上传控件等多种形式。表 1.1 说明了<input>标签内 type 属性的可能取值及对应的控件类型。

表 1.1 <input>标签内 type 属性的可能取值及对应的控件类型

控 件 类 型	type 属性的取值	控 件 类 型	type 属性的取值
单行文本输入控件	text	普通按钮	button
密码输入控件	password	提交按钮	submit
复选框	checkbox	取消按钮	reset
单选按钮	radio	图片提交按钮	image
隐藏输入控件	hidden	文件上传控件	file

(2) 下拉列表框控件

Web 应用系统的开发者利用下拉列表框控件可以为用户提供多种可能的选择型输入信息，用户根据自己的需要选择其中的某个项目。在 HTML 页面中利用<select>标签(其中的 name 属性定义该下拉列表框的对象名称)可以产生出下拉列表框，而利用包含在其中的<option>标签(其中的 value 属性指示该项被选中后向服务器程序提交的具体值，selected 属性表示该选项为默认选中的项目)定义出下拉列表框中的每个列表项目。

下面为利用<select>和<option>标签组合定义某个表单中下拉列表框的代码片段示例，其中的第一个列表项目作为默认的选中项目(注意黑体所标识的标签)。

```
<select id="showStyle_FormSelect" >
    <option value="0" selected>选择风格</option>
    <option value="1">正常风格</option>
    <option value="2">古典风格</option>
    <option value="3">自定义风格</option>
</select>
```

(3) 多行文本输入控件

由于<input type="text">标签只能产生单行文本输入控件，而在许多 Web 应用系统(比如 BBS 论坛、新闻等系统)中需要收集用户输入的多行文本信息，此时可以在<form>表单标签中内嵌<textarea>标签。比如代码<textarea name="bbsContent" cols="60" rows="20" disabled ></textarea>将创建出一个多行文本输入控件。

<textarea>标签中的 name 属性定义该多行文本输入控件的标签对象名称，cols 属性定义列宽，rows 属性定义行高。如果添加了 disabled 属性，则该多行文本输入控件将被改变为“只读”效果，操作者不能编辑修改其中的文本内容。

9. Web 表单及相关的各个控件标签的应用示例

由于 Web 表单是 Web 应用系统中的主要用户交互界面控件，为了能够让读者迅速掌握 Web 表单及相关的各个控件标签的应用，在例 1-4 中给出了一个 Web 应用系统中用户注册功能的页面表单标签代码，但作者对它进行了简化处理并省略了其中的各种样式定义控制的相关内容，并且采用表格直接排版该表单中的各个控件元素。

例 1-4 体现 Web 表单及相关的各个控件标签的应用示例

```
<form action="#" method="post"  id="userRegisterInfoForm">
    <table>  <tr><td align="right">用户名称：</td>
                 <td><input type="text" id="userName"></input></td></tr>
             <tr><td align="right">用户密码：</td><td><input type="password"
                       id="userPassWord"></input></td></tr>
              <tr><td align="right">确认密码：</td><td><input type="password"
                       id="userPassWord"></input></td></tr>
             <tr><td align="right">选择身份类型</td><td>
                   <select id="userType">
                       <option value="1" selected>普通用户</option>
                       <option value="2">VIP 用户</option>
                       <option value="3">版主</option>
                       <option value="4">管理员</option>
                   </select></td></tr>
             <tr><td align="right">你的性别：</td><td><input type="radio"
                         id="sex" checked>男</input>
                        <input type="radio" id="sex">女</input></td></tr>
             <tr><td align="right">你的爱好：</td><td>
                   <input type="checkbox" id="intersting" checked>读书</input>
                   <input type="checkbox" id="intersting">运动</input>
                   <input type="checkbox" id="intersting">爬山</input>
                   </td></tr>
             <tr><td align="right">请选择你的头像：</td><td><input type="file"
                       id="userImage" ></input>  </td></tr>
             <tr>  <td align="right">你的建议：</td><td><textarea id="advise"
                       col="20" row="10"></textarea></td></tr>
             <tr><td align="right">
                       <input type="submit" value="确认"></input></td>
                 <td><input type="reset" value="取消"></input></td></tr>
     </table></form>
```

例 1-4 所示的代表用户注册功能的页面表单在浏览器中本地浏览的最终显示效果如图 1.6 所示。另外，请注意在 HTML 规范中，各种<input>类型的标签不是必须需要结束标签的；而在 XHTML 规范中，所有的<input>标签都必须要有成对的结束标签（被正确地关闭）。例 1-4 中的标签是严格按照 XHTML 的语法要求设计的，所有的标签都有对应的结束标签。

10. 利用<fieldset>标签实现对 Web 表单中的项目分组

在 Web 表单设计中，经常需要对表单中的特定项目分组以体现数据之间的内在关系，这可以利用<fieldset>标签来实现。<fieldset>标签是要成对出现的，并且在一个 Web 表单中可以包含多个成对的<fieldset>标签，以达到不同形式的分组设计效果。

但每对<fieldset>标签定义一个分组，对每组的内容描述可以使用<legend>标签加以定义说明，<legend>标签必须是<fieldset>标签内的第一个子标签，<legend>标签定

图 1.6　例 1-4 所示的页面表单在浏览器中的显示效果

义的内容一般显示在分组框的左上角。下面为利用<fieldset>和<legend>标签组合定义表单中某个数据分组的代码片段示例。

```
<form>  <fieldset><legend>个人信息</legend>
    <label for="userName">姓名:<input type="text" id="userName"/></label>
    <label for="userAge">年龄:<input type="text" id="userAge"/></label>
</fieldset></form>
```

上面的代码片段示例在浏览器中的最终显示效果如图 1.7 所示，但在示例中省略了对该表单中各个控件的样式定义文件。

图 1.7　利用<fieldset>和<legend>标签实现对表单中项目分组的示例图示

1.2　应用 Dreamweaver 实现可视化页面设计

1.2.1　网页设计工具软件 Dreamweaver

1. Dreamweaver 是一款专业的 Web 客户端和服务器端页面开发工具程序

该工具软件可以用于对 Web 站点、Web 页面和 Web 脚本程序进行设计、编码和开发实现，并且提供可视化的编辑环境和良好的及时动态帮助信息。Web 页面开发人员通过应用 Dreamweaver 工具软件中的可视化编辑功能，可以快速地创建页面内容而无须编写任何的

HTML 标签代码。当然，开发者也可以手写标签，直接编码实现对页面中的内容描述。因为在 Dreamweaver 程序中，为开发人员提供了两种不同形式的开发视图：代码和设计。

Dreamweaver 工具软件是曾经风靡一时的网页三剑客成员之一的页面可视化开发工具，早期属于 Macromedia 公司，但现在已经属于 Adobe 公司。目前版本的 Dreamweaver 程序，不仅支持 Web 客户端的页面开发所涉及的各种技术（比如 XHTML、CSS 和 JavaScript 等），也支持各种主流的 Web 服务器端的开发技术及编程语言，例如微软早期的 ASP 以及现在的 ASP. NET、ColdFusion 标签语言（ColdFusion Markup Language，CFML）、Sun 公司的 JSP、Rasmus Lerdorf 创建的 PHP 等目前比较流行的进行动态网站开发的编程语言。

下文将介绍如何应用 Dreamweaver 工具程序进行 Web 项目中各个页面的设计和实现。Dreamweaver 程序操作简单，基本上类似于微软公司的 Word 软件。

2. 启动 Dreamweaver 工具软件程序

读者可以根据自己所获得的 Dreamweaver 工具软件的版本首先在本机中安装该软件，目前也提供有免费安装的版本（俗称“绿色版”）。作者采用的就是免费安装的版本，直接单击其中的 Dreamweaver. exe 程序文件名就可以启动 Dreamweaver 工具软件，程序主界面如图 1.8 所示（该图为主界面的局部截图）。



图 1.8　Dreamweaver 工具软件启动后的程序主界面

3. 在 Dreamweaver 工具软件程序中新建 Web 站点

（1）新建和命名 Web 站点

Dreamweaver 程序对页面的组织和管理是应用“Web 站点”方式实现的，基本上类似于 Java 程序开发中 Eclipse IDE 工具中的项目（Project）的概念。因此，Web 开发人员首先需要在 Dreamweaver 工具程序中为项目新建一个 Web 站点。

单击 Dreamweaver 工具程序主窗口内主菜单栏中的“站点”菜单，然后再单击其中的“新建站点”子菜单项，出现如图 1.9 所示的新建站点对话框。在该对话框中输入将要创建站点的名称（本示例为 WebCRM）以命名该 Web 站点。



图 1.9　新建和命名本项目的 Web 站点

（2）为本 Web 站点选择对应的服务器端开发实现技术

单击“下一步”按钮后，将出现如图 1.10 所示的选择服务器端开发实现技术对话框，在该对话框中为本 Web 站点选择对应的服务器端开发实现技术，本示例选择 JSP 类型。



图 1.10　选择服务器端的实现技术为 JSP

（3）选择 WebCRM 项目的 Web 站点根目录位置

单击“下一步”按钮后，将出现如图 1.11 所示的对话框，在该对话框中根据本 Web 项目的页面文件实际存放的磁盘目录选择项目的 Web 站点根目录的磁盘路径位置，本 WebCRM 示例项目在作者的机器中为 F:\WebCRM 目录。



图 1.11　选择项目的 Web 站点根目录位置

继续单击“下一步”按钮，将进入新建 Web 站点的其他配置信息对话框。对这些配置信息不需要改变，直接选用 Dreamweaver 程序中所提供的默认选择项目即可，一直单击各个对话框中的“下一步”按钮，直到出现包含有“完成”按钮的对话框（在此省略了对这些步骤的截图）。最后单击“完成”按钮，将创建出本 Web 站点，目录结构如图 1.12 所示。

4. 在 Web 站点中新建项目中各个页面文件的目录

由于在实际项目中存在大量的信息，一般都应该把这些信息内容分类存放以便于以后的维护修改。因此，需要在本 Web 站点中建立各种通用的文件目录，比如 CSS 目录（存放项目中的各个样式表文件）、images 目录（存放项目中的各种图片文件）、flash 目录（存放项目中的各种 Flash 动画文件）、javascript 目录（存放项目中各个客户端的 JavaScript 脚本程

序)等。

当然,除了这些通用的文件目录以外,开发人员也可以根据项目需要建立其他的目录。如图 1.12 所示为某个客户关系信息管理系统(Customer Relationship Management,CRM)中各个目录的局部截图。

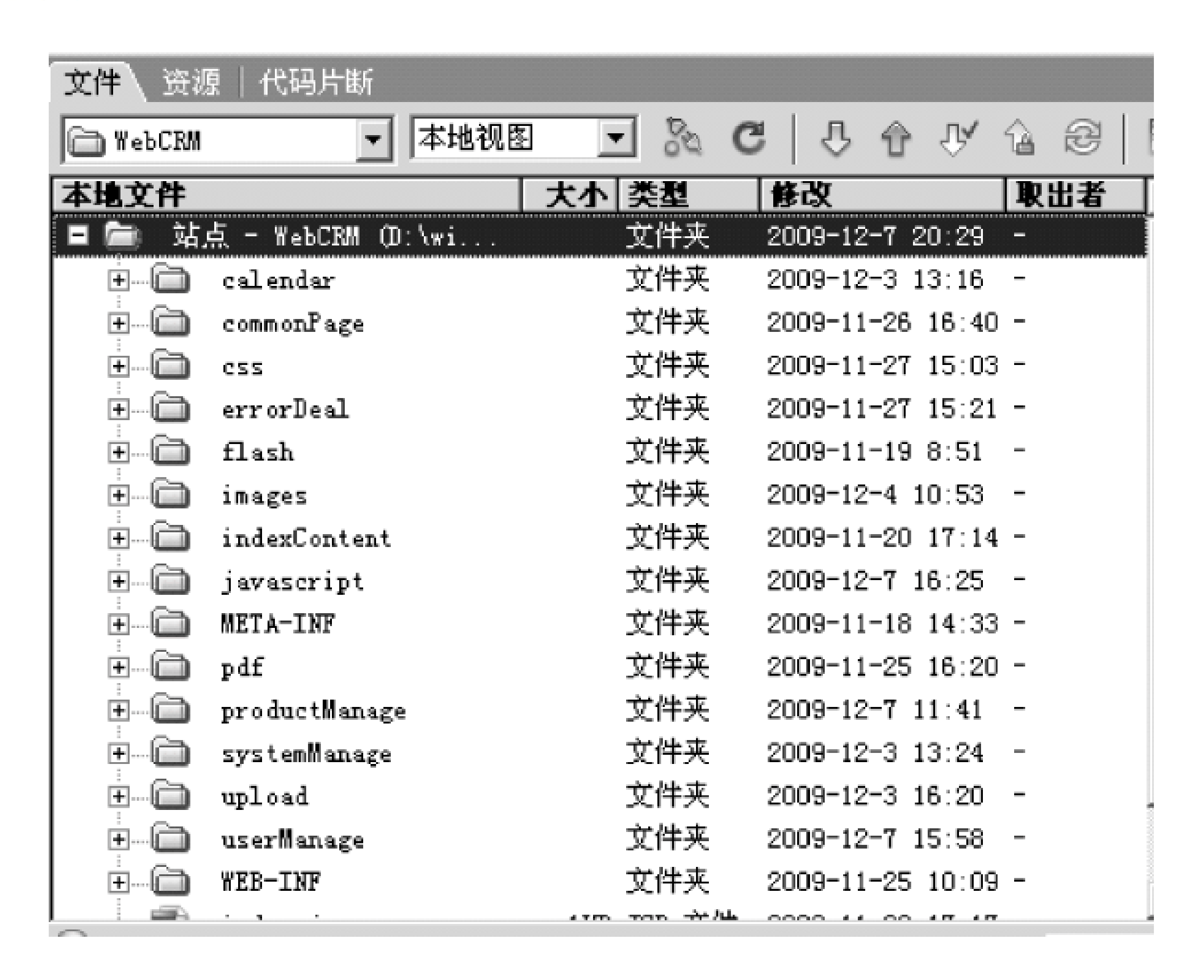


图 1.12　某个客户关系信息管理系统的文件目录结构图示

1.2.2　客户关系管理信息系统页面设计和实现

1. 新建 CRM 系统中 Web 站点的首页面文件

(1) 设置 Dreamweaver 页面编辑器的默认编码

由于 Dreamweaver 软件支持多种不同的语言,在项目的页面设计中为了能够正确地保存页面中的中文信息,首先应该设置页面编辑器的默认编码为中文编码。选择“文件”主菜单(如图 1.8 所示)中的“新建”子菜单项,弹出一个“新建文档”对话框,如图 1.13 所示。在该对话框中单击“首选参数”按钮,将弹出如图 1.14 所示的“首选参数”对话框。

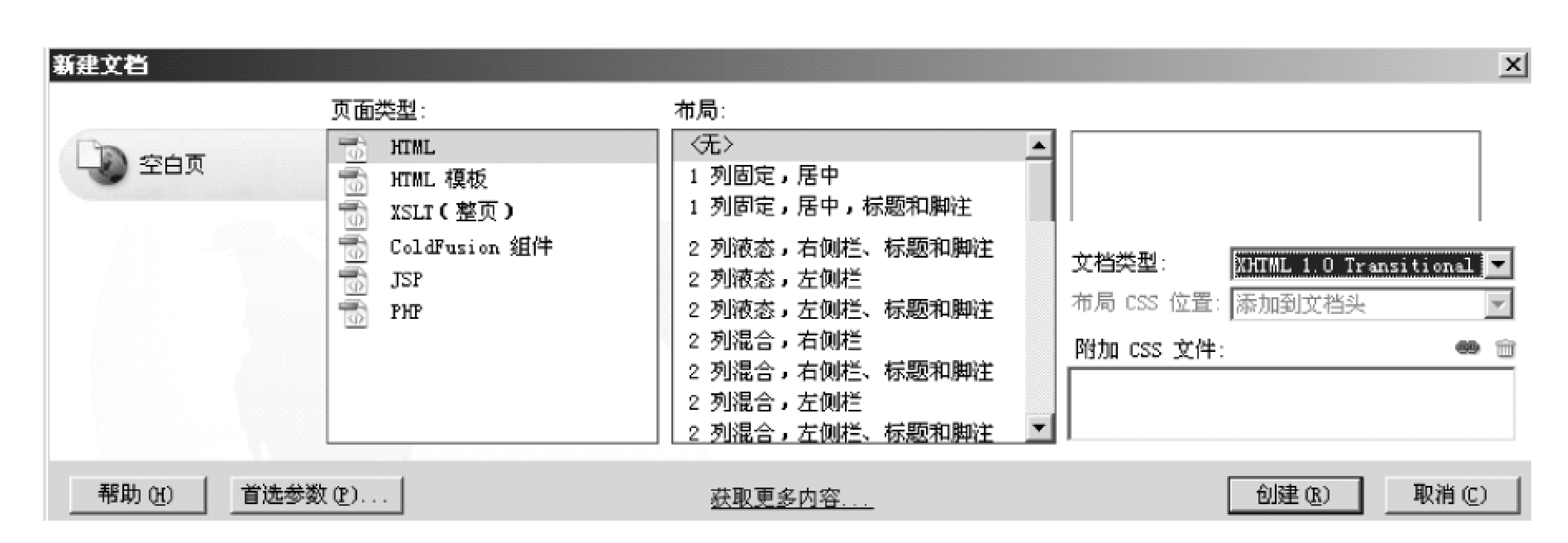


图 1.13　“新建文档”对话框

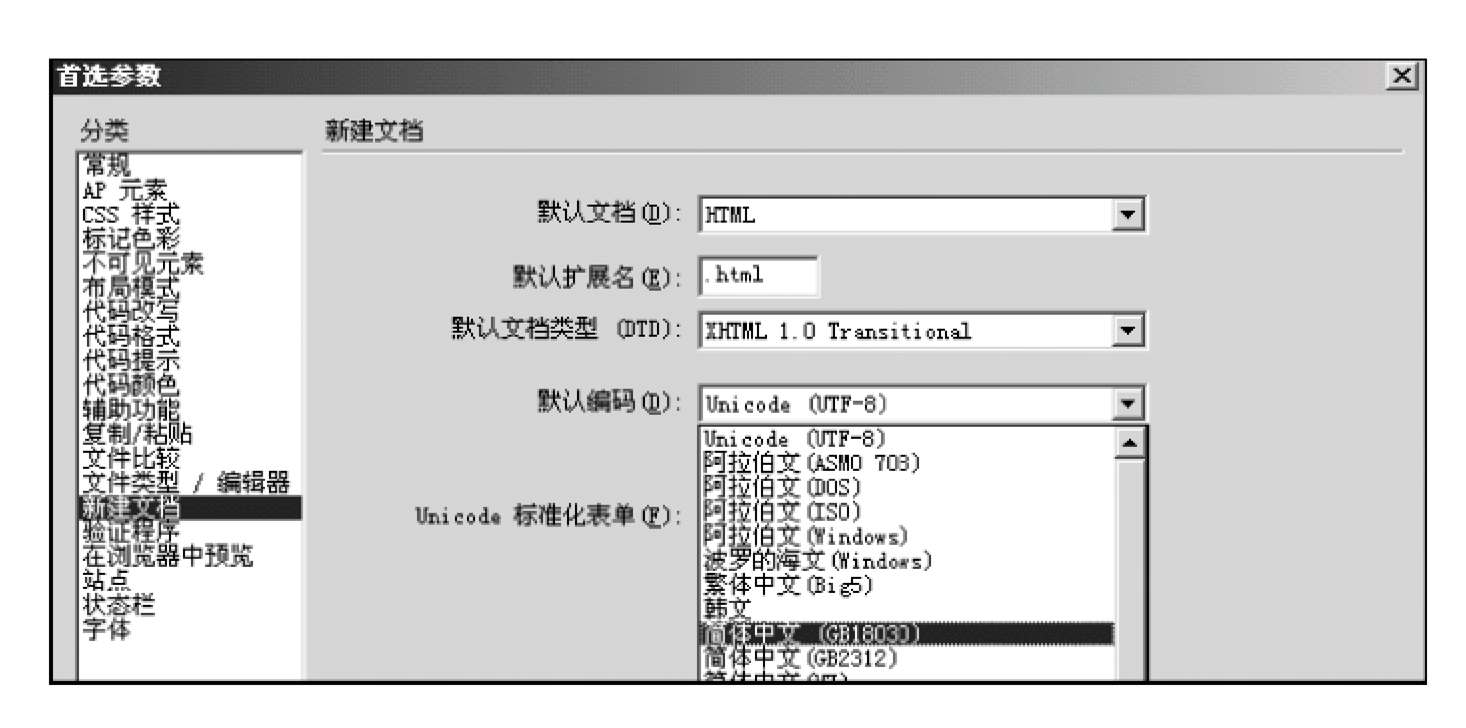


图 1.14　设置页面的默认编码为简体中文编码

在如图 1.14 所示对话框左面的“分类”中选择“新建文档”类型，然后在右面的“默认编码”下拉列表框中选择“简体中文(GB18030)”项目。单击“完成”按钮，就可以改变 Dreamweaver 页面编辑器的默认编码为中文编码。

(2) 新建项目中的首页面文件

选择“文件”主菜单中的“新建”子菜单项，弹出如图 1.13 所示的“新建文档”对话框。在“页面类型”栏中选择“HTML”类型，然后在“布局”栏中选择“无”(表示不需要预先进行页面的布局设计)，最后单击“创建”按钮，将创建出一个空的 HTML 页面。

命名该 HTML 页面文件名为 index.html(目前暂时采用客户端 HTML 页面)，并将该 index.html 文件保存到 Web 项目的站点根目录中。

2. 为项目中的首页面文件添加<head>标签中的子标签

(1) 在 index.html 页面中添加<head>标签中的各种子标签

在页面文档头部(由成对的<head>标签定义)中可以放置的子标签主要有<title>、<meta>、<link>、<style>、<script>、<base>等。

其中成对的<title> 标签用于定义网页的标题，并且所定义的标题文字会显示在浏览器窗口的标题栏中。本示例 index.html 文件的<head>标签中的各个子标签如例 1-5 所示。

例 1-5　index.html 页面的部分标签示例

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>  <head><title>蓝梦集团 CRM 系统首页</title>
    <meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta name="keywords" content="蓝梦集团,CRM,信息管理系统">
    <meta name="author" content="杨少波,联系邮件 abc@abc.com.cn">
    <meta name="description" content="这是蓝梦集团 CRM 系统">
    <meta content="index,follow" name="robots">
    <link rel="stylesheet" type="text/css" href="styles.css">
```

在搜索引擎中搜索关键字的描述文字

在搜索引擎中所显示出的描述文字

```
</head>
<body>这是蓝梦集团 CRM 系统的首页内容</body></html>
```

(2) HTML 中的<meta>标签主要有两种形式

页面文档头部中的各种<meta>标签主要有两种形式：<meta name="名称" content="该名称对应的值" /> 和<meta http-equiv="属性项目名" content="该属性项目对应的值" />，并且可以包含多个不同的<meta>标签，书写的顺序可以任意。

(3) 带有 name 属性的各种<meta>标签

它同样也是说明性标签，并对该页面的最终显示效果没有任何影响。其中的 name 属性定义了该<meta>标签的性质，而 content 属性定义了该标签的值。name 属性的主要取值及对应的功能说明如下。

- 关键字(keywords)：指定与站点有关的关键字，给搜索引擎提供分类信息。
- 描述(description)：为搜索引擎提供 Web 站点内容简介的描述信息。
- 作者(author)：注明网页的设计实现者，通常是作者姓名或联系方式。
- 搜索机器人(robots)：当一个搜索机器人(也称为搜索蜘蛛)程序访问一个 Web 站点时，它首先爬行该站点根目录并检查其下是否存在 robots.txt 文件，如果存在该文件，搜索机器人程序就会按照该文件中的描述内容来确定搜索的范围；如果该文件不存在，搜索机器人程序就沿着页面中的各个超链接顺序抓取信息。

(4) <meta name="robots" content="属性值">标签中的 content 属性

搜索引擎(Google 的搜索引擎为 Googlebot，百度的搜索引擎为 Baiduspider)自动检索页面中<meta> 标签的 keywords 和 description 属性，并将它们的信息内容加入到索引数据库中，然后再根据关键词被检索的密度排序网站。因此，为了使得本 Web 站点能够更好地被搜索引擎程序搜索到，Web 系统的开发人员需要合理地设置<meta> 标签的 content 属性的值。如图 1.15 所示为百度搜索引擎搜索"蓝梦集团"关键字后的搜索结果(局部截图)。

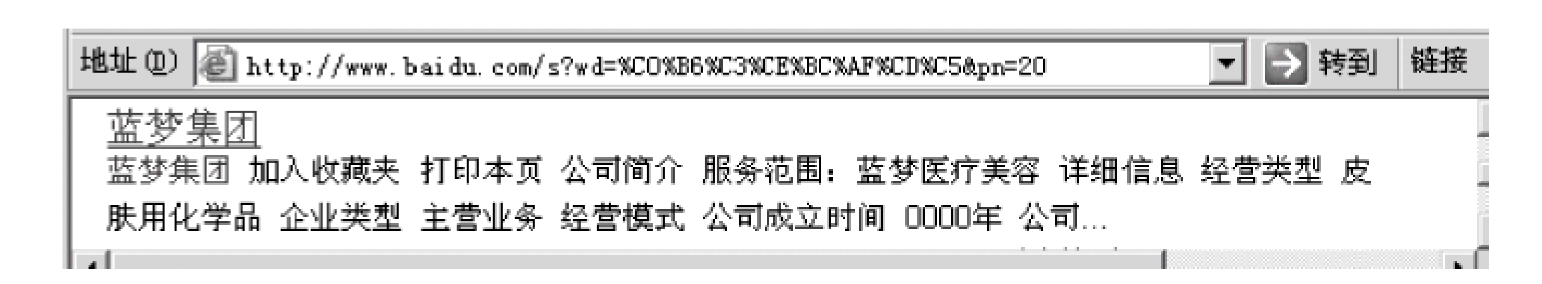


图 1.15　通过百度搜索"蓝梦集团"关键字后的搜索结果

由于搜索引擎本身是一个自动化的程序，并根据特定的搜索策略和算法不断地访问 Internet 上各个 Web 站点的网页信息并建立索引数据库，然后再对信息进行组织和处理，并将处理后的最终结果信息显示给用户。搜索引擎是为用户提供检索服务的系统，使访问者能在某个搜索引擎中快速地搜索到目标 Web 网站的有关网页。

<meta name="robots" content="属性值">标签中 content 属性的可能取值及对应的功能说明如下：

- content="all"：页面文件将被检索，并且页面中的超链接可被查询。
- content="none"：页面文件不被检索，也不需要查询页面中的各个超链接。

- content="index"：页面文件将被检索。
- content="follow"：查询页面中的各个超链接。
- content="noindex"：页面文件不被检索，但可被查询超链接。
- content="nofollow"：页面文件不被检索，且不可查询页面中的各个超链接。

(5) 带有 http-equiv 属性的各种<meta>标签的作用

这些<meta>标签是功能性标签，它为浏览器设置一些工作参数以帮助浏览器正确和精确地显示出本网页的内容。因此，它对网页在浏览器中的最终显示效果是有一定影响的。http-equiv 属性的可能取值及对应的功能说明如下。

- 字符集(Content-Type)：指定网页中使用的字符集，对于中文页面一定要设置为中文编码(GB2312 或者 GBK 等形式)，否则会出现中文乱码现象。因为如果在一个网页中没有指明字符集，浏览器会应用默认的字符集显示该网页的内容，如果它和网页中实际使用的字符集不一样时，就会使得网页信息在显示时出现乱码。
- pragma：禁止浏览器对本页面内容进行缓存处理。浏览器提供缓存技术本来是为了提高访问性能，以避免重复地请求相同的 Web 页面。但在许多需要实时信息显示的 Web 应用系统中，则不能让浏览器缓存本页面的内容信息。当然，此时用户也将无法用脱机形式浏览本页面的内容信息。
- window-target：用于指定显示页面的浏览器窗口。
- refresh：让浏览器按照所设置的时间间隔自动地进行刷新或者请求其他的目标资源。比如：<meta http-equiv="refresh" content="5 " />标签表示每隔 5 秒钟(由其中的 content 属性定义)让浏览器重新请求当前的页面。
- expires：指定网页在缓存中的过期时间，一旦网页过期，必须要从服务器中重新下载本页面的信息内容。比如：<meta http-equiv="expires" content="0" />标签同样也能够达到禁止浏览器对本页面内容进行缓存处理的效果。

(6) link 链接标签的作用

<link>标签主要用于定义本页面文档与外部其他资源文件的引用关系，其中最常见的用途是指示本页面所需要的 CSS 样式表文件，例 1-5 中的<link>标签就是指示本页面所需要的外部样式表文件 styles. css。

3. 利用可视化方式为系统添加全局导航菜单栏

(1) 在项目的首页面中添加全局导航菜单栏的布局表格

选择 Dreamweaver 工具软件中“插入记录”主菜单中的“表格”子菜单项，弹出如图 1.16 所示的“表格”对话框。在该对话框中设置表格的“行数”为 1，“表格宽度”为 100%(采用相对比例)，“边框粗细”为 0(表示不需要边框线)，然后单击“确定”按钮，将在当前页面中创建出指定的表格(在本示例中没有应用<div>标签实现对全局导航菜单栏的布局定位，而是简单地应用表格，主要是为了说明 Dreamweaver 工具软件的使用方法)。

(2) 在全局导航菜单栏的表格中添加各个超链接文字和链接到目标资源文件

首先添加各个超链接的说明文字，然后分别为各个说明文字提供超链接的目标资源文件的定义，这可以借助于 Dreamweaver 工具软件中提供的【属性】面板以可视化方式操作实

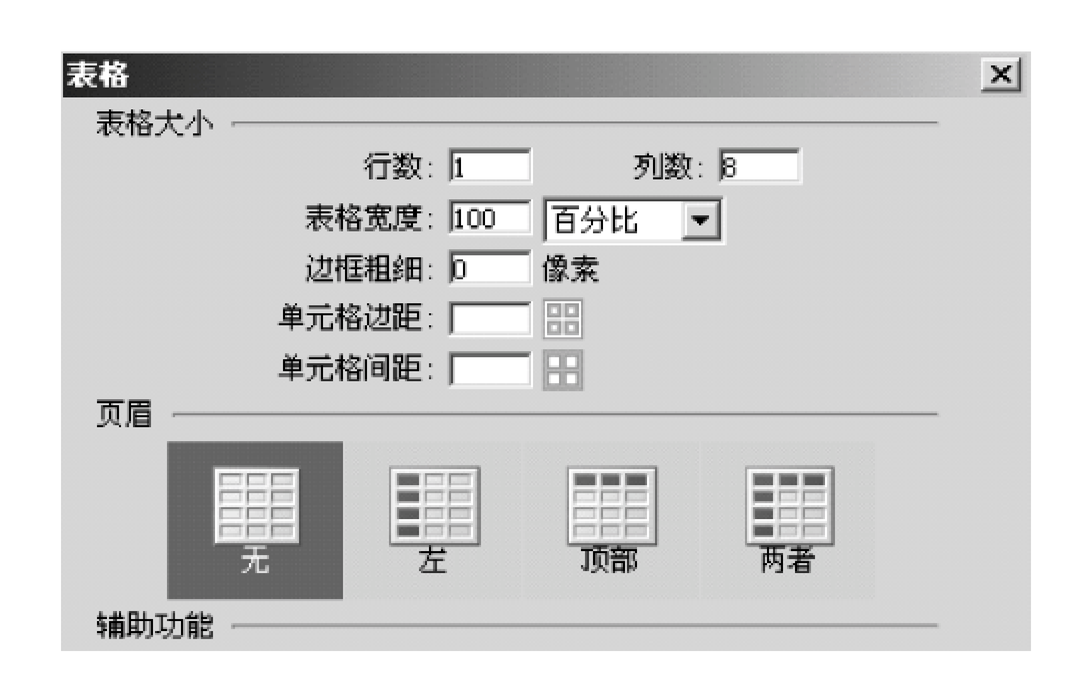


图 1.16 “表格”对话框中的各个设置项目

现,如图 1.17 所示。



图 1.17 利用“属性”面板以可视化方式创建各个超链接

(3) 为全局导航菜单栏添加各种样式定义以产生特定的效果

在 Dreamweaver 工具软件中新建 CSS 样式表文件,并设计各个样式项目以进一步格式化全局导航菜单栏和其中的各个超链接的显示风格。本示例的最后设计效果如图 1.18 所示(局部截图)。



图 1.18 为全局导航菜单栏添加样式定义后的显示效果图

4. 利用可视化方式为系统添加各种功能性表单

(1) 在用户登录功能页面中添加登录表单

选择 Dreamweaver 软件中“插入记录”主菜单中的“表单”子菜单项,将自动地在当前页面中添加一个<form>标签,然后设置该<form>标签的 id 属性为表单的名称、method 属性为表单提交的方式(可以为 POST 或者 GET 方式之一)、action 属性为请求的服务器端的目标 JSP 页面文件或者 Servlet 程序的 URL 地址等。

(2) 在登录表单中添加各个输入控件标签以构建出本表单的操作界面

插入了<form>表单标签后,就可以为该表单添加内部的输入控件了。在 Dreamweaver

工具软件的“插入记录”主菜单中，只需选择“表单”子菜单内对应的控件子菜单项即可，如图 1.19 所示为各个子菜单项(局部截图)。



图 1.19 Dreamweaver 所支持的与表单有关的各个子菜单项

在登录表单中分别添加验证码、用户名、密码的单行输入框控件和用户类型的下拉列表框控件，为了使得表单中的各个输入控件能够整齐地对齐，可以应用表格布局定位排列它们。最终的设计效果如图 1.20 所示。



图 1.20 CRM 系统中用户登录表单的预览效果

对 CRM 系统中其他功能性页面的设计和实现也基本上与用户登录功能页面的设计和实现类似，只是表单内的输入控件和 CSS 样式定义等有差别。

5. 利用可视化方式为系统中的页面进行布局定位

(1) 利用<div>标签为 index.html 首页面内容进行布局定位

根据用户方的需求，CRM 系统首页面内容的布局从上到下主要分为如下几个版块区域：页头区(包含企业的 Logo 图像等信息)、全局导航菜单栏及各个超链接，中间的页面内容区和底部的版权信息区。根据此界面需求，开发人员在 Dreamweaver 工具软件中利用<div>标签为 index.html 首页面内容进行布局定位。在 Dreamweaver 工具软件的“插入记录”主菜单中，选择“布局对象”子菜单内的“Div 标签”子菜单项，系统将弹出“插入 Div 标

签”对话框。在“插入 Div 标签”对话框的“ID”输入框中输入<div>标签的 ID 属性值(本示例为 topLogoBar),如图 1.21 所示。



图 1.21　利用可视化方式添加实现布局定位的<div>标签

然后依此方式再添加代表其他区域的<div>标签,并分别设置 id 属性值以命名各个<div>标签,最终产生出如下例所示的各个<div>标签。

```
<body><div id = "topLogoBar"></div>
        <div id = "navMenuBar"></div>
        <div id = "centerPageContent"></div>
        <div id = "pageFooter"></div></body>
```

(2) 将页面中各个大的区域再进一步细化为若干小的块区域

为了能够精确地定位图片、文字、表格和超链接等内容信息,需要将页面中总体布局中的各个大的区域再进一步细化为若干小的块区域。这只需要在对应的<div>标签中再插入子<div>标签即可,并且可以多级嵌套,从而最终在横向和纵向方向上不断地细化、分割某个大的块区域。如图 1.22 所示为 CRM 项目的页脚区域内版权信息的布局定位(局部截图)。

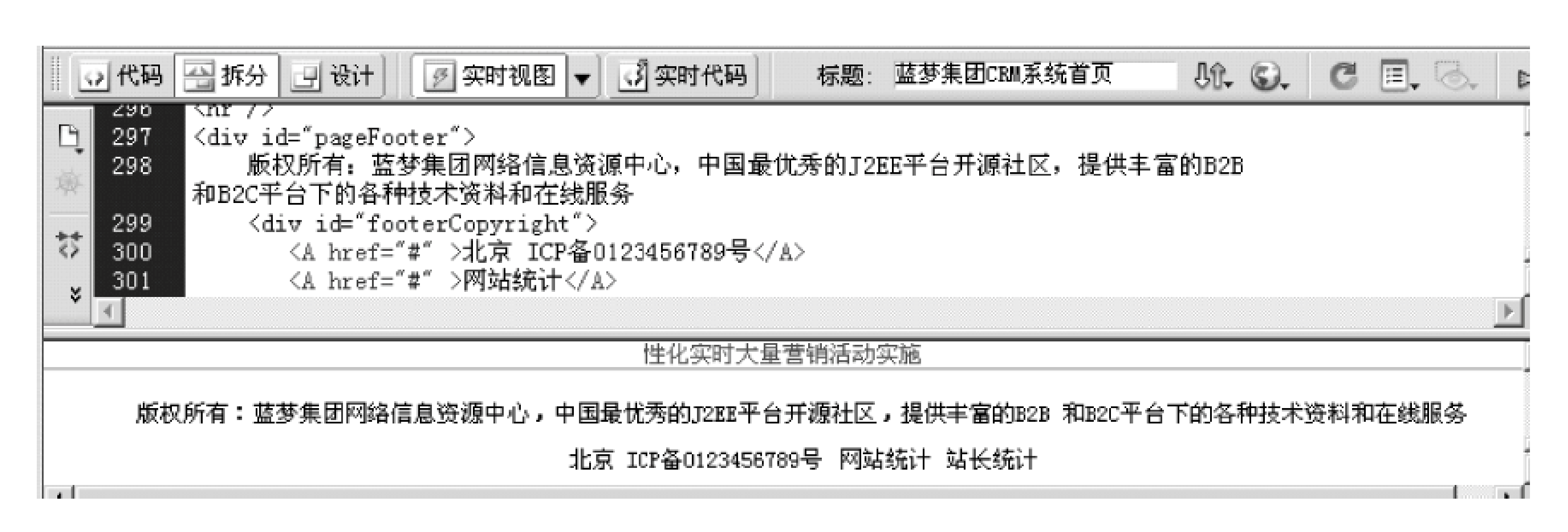


图 1.22　CRM 项目的页脚区域中版权信息的布局定位

6. 为各个布局<div>标签添加对应的 CSS 样式定义

随着对本教材的不断深入学习,读者在熟悉和掌握了与 CSS 样式表有关的知识后,就可以为各个布局<div>标签添加对应的 CSS 样式定义了。在本教材的第 4 章“应用 XHTML+CSS+Div+JS 构建网站”中将会详细地介绍这些方面的知识和相关的实现技术。本示例在添加了 CSS 样式和完善了页面内容后的最终结果如图 1.23 所示(局部截图)。



图 1.23　为各个布局<div>标签添加对应的 CSS 样式后的结果

1.3　XHTML 标签语言及应用

1.3.1　W3C 发布的 XHTML 标签语言及规范

1. XHTML 是什么形式的标签语言

XHTML 本身并不是什么新的标签语言，它其实是更"规范"的 HTML 版本。可以这样理解 XHTML 标签语言："XHTML = HTML + XML"，也就是满足"XML 语法"要求的 HTML 页面。

W3C(World Wide Web Consortium，万维网组织)发布 XHTML 规范的主要目的是希望 Web 开发人员能够设计并实现比较"严谨"和"规范"的 HTML 页面，也就是要求 Web 页面设计者能够按照 XML(eXtensible Markup Language，可扩展的标签语言)的"语法规范"进行 Web 页面设计，以避免浏览器在显示时产生"二义性"，并保证在各个不同的浏览器中能够实现一致的显示效果。

2. XHTML 标签语言的基本语法要求

Web 页面设计人员在应用 XHTML 开发 Web 页面时，不仅要遵守 HTML 语法的基本要求，也要遵守与 XHTML 有关的如下附加要求：

- 所有的标签元素必须能够被正确地嵌套和有结束标签。
- 标签名必须用小写字母。
- XHTML 文档必须拥有根标签。
- 属性名必须小写，并且必须加引号，且属性名不能采用简写形式。
- 用 id 属性名代替 name 属性名。
- XHTML 页面中的 DTD(Document Type Definition，文档类型定义)定义页面文件中各个标准的 HTML 标签元素及属性的语法规则。

3. W3C 为什么要放弃 HTML 标签语言

(1) 首先，HTML 只能在 Web 浏览器中显示和应用

随着 Internet 网络技术的不断发展，涌现出了许多手持形式的网络客户端设备，比如手

机、PDA、信息家电等都不能直接显示由 HTML 所标识的信息。因此，HTML 不能适应现在越来越多的网络设备和应用的需要。

（2）其次，HTML 标签本身是不规范和比较臃肿的

由于 W3C 当时在制定 HTML 规范时不是一个“强制性”的标准规范，从而导致目前不同厂商的 Web 浏览器和不同版本的浏览器对同一个 HTML 标签的支持也是不一致的。许多厂商为了提升浏览器的性能，创建了自己的 HTML 扩展子集；另外，Web 页面开发者在应用 HTML 标签时也不严格遵守 HTML 的规范要求而“随意”地编写 HTML 标签。

这种状况的最终结果将导致 Web 浏览器程序需要足够的“智能判断”、“程序庞大”和“程序复杂”才能够正确地显示出 HTML 页面；另外，设计和开发实现能够在所有的 Web 浏览器上都能一致性地显示 HTML 页面其实也是十分困难的，甚至是不可能的。这为 Web 应用系统的兼容性设计和实现带来一定的复杂性和人为地增加了开发实现的工作量。

（3）另外，HTML 标签是将“内容”和“表现”混杂在一起

任何标签所描述的“内容”发生改变、页面“结构”的局部调整变化，都会影响到整个页面“表现”部分的标签，也都需要进行相应地修改或者调整，而且是一点一点、细致地修改。而 XML 技术的可扩展性不仅允许 Web 开发者自己定制体现结构的标签，并且还提供了多种不同的显示风格控制技术，比如应用 CSS 和 XSLT（eXtensible Stylesheet Language Transformation，XSLT 是一种用来转换 XML 文档结构的语言）对页面内容进行显示格式控制，最终实现将 Web 页面中的“内容”和“表现”相互分离。

（4）最后，还表现在重用页面内容时也比较困难

正是由于 HTML 标签是将“内容”和“结构”与“表现”混合在一起的，所以重用同一个页面的内容也比较困难。必须将“内容”和“结构”与“表现”相分离，才能有利于对页面“内容”的重用，以及对同一个页面“内容”附加不同的“表现”而产生出“换皮肤”的显示效果。

因此，有必要规范 HTML 标签本身以及对 HTML 的应用方式，也就是制定出强制性的标准和规范。Web 标准化本身和设计的 Web 文档的标准化给 Web 开发人员带来的好处不仅体现在页面结构清晰上，而且也能够提高页面的可读性和可维护性、降低客户端浏览器的版本升级和改变浏览器时可能带来的兼容性方面的成本开销；另外，由于页面“内容”和“表现”两者相互分离，更易于对页面“内容”的维护和“表现”形式的扩展；还有，标准化的 Web 文档也更能够被搜索引擎搜索命中；最后，由于页面“内容”和“表现”相互分离，页面“内容”能够被更多的设备访问，如打印机、手机等设备。

4. W3C 为什么要推荐 XML 和 XHTML 标签语言

（1）XHTML 代表更规范和更严谨的 HTML

W3C 首先制定了 XML 的基本规范，并不断地推广 XML 技术的应用。目前 XML 的可扩展性广泛地应用在不同平台的数据交换、网络通信协议等领域，在 Java 及 J2EE 技术平台中还广泛地应用 XML 文档作为项目的系统配置文件。

但 XML 的可扩展性（也就是允许 XML 使用者自己定义标签和属性的名称）为浏览器程序的设计和开发实现也带来了许多不便——没有标准的标签名和属性名，浏览器程序也就无法识别和理解不同用户自定义的各种 XML 标签的“表现”要求。

为此，W3C 又制定了 XHTML 技术标准和相应的技术规范。因此，XHTML 不仅是将“随意”的 HTML 的应用向“规范”和“严谨”的 XML 技术过渡的一个桥梁，同时也是一个引导 Web 页面设计人员逐步走向规范、走向 XML 的过渡技术实现方案。尽管在 XHTML 的名称中也有“X”，但它并不能扩展(自定义标签名)，只是在 HTML 规范的基础上要求 Web 页面设计人员一定要严格遵循和应用 XML 的技术规范设计和开发实现 Web 页面。

(2) 提高 HTML 页面信息的可重用性

在应用中经常需要在 XML 文档中重用 HTML 页面中的标签信息，但由于普通的 HTML 并不是基于 XML 语法格式的，无法直接重用；必须要对 HTML 文档进行修改或者转换后成为正确的 XML 文档，才能被重用。

XHTML 由于本身就是 XML，因此在 XML 文档中可以直接包含 XHTML 片段，同样在 XHTML 文档中也可以包含 XML 片段。

(3) 保证相同的 HTML 页面内容在不同的浏览器中显示相同的效果

浏览器在解析不规范的 HTML 页面时，首先必须要修正它。而不同厂商的浏览器(而且目前的浏览器除了 PC，还有 PDA、手机、电视等设备中的浏览器)的修正方式并不一致，这就有可能出现相同的 HTML 页面内容在不同的浏览器中显示出不同的效果。

5. 在开发中如何验证页面中 HTML 标签的有效性

Web 开发人员在应用 XHTML 技术时，如果没有严格遵守 XHTML 的规范要求，XHTML 的解析程序将“拒绝”解析不规范的 XHTML 文档。在 Dreamweaver 工具程序中提供有对基于 XHTML 页面(也包括 CSS)的有效性检查功能，只需要选择“窗口”主菜单中的“代码检查器”子菜单项，Dreamweaver 工具程序就会自动地扫描该 XHTML 页面文件中的各个标签是否满足 XHTML 语法的基本要求。如图 1.24 所示为对某个不规范的 XHTML 页面检查后的结果图示，本例中故意将页面中的＜title＞标签的开始标签和结束标签不一致(没有严格遵守字符大小写的一致性要求)，Dreamweaver 工具程序在代码检查器窗口中准确地反映出这个错误。



图 1.24　Dreamweaver 对某个不规范的 XHTML 页面检查后的结果图示

6. 应用 W3C 的标签有效性在线服务检查发布后的系统有效性

(1) W3C 目前所提供的各种在线验证服务

W3C 是一个专注于“领导和发展 Web 技术”的国际工业行业协会，为了促进 XHTML 的规范应用，为 Web 开发人员提供一个在线免费的标签有效性服务(Markup Validation Service)，帮助 Web 页面设计人员检验页面中各个标签的有效性，从而改善 Web 应用系统

的品质。W3C 目前所提供的在线验证服务主要包括以下几项。

- 标签有效性服务：可检验 HTML、XHTML、SVG 或 MathML 格式的网页，网址为 http://validator.w3.org/。
- 超链接有效性服务：可检验 HTML/XHTML 网页中是否有无效的超链接存在，网址为 http://validator.w3.org/checklink。
- CSS 有效性服务：可检验 CSS 样式表或者使用 CSS 的网页是否有效，网址为 http://jigsaw.w3.org/css-validator/。

(2) 应用 W3C 的标签有效性在线服务

首先进入 W3C 的标签有效性服务网站(http://validator.w3.org/)，然后输入待检查的目标网站的 URL 地址，并单击页面中的 Check 按钮，系统将自动地检测各个标签、源代码、HTML/XHTML 文档等是否存在语法错误并提出修改建议。如图 1.25 所示为对某个著名门户网站检测的结果(局部截图)，反映出有 12 个错误和 1 个警告性提醒。

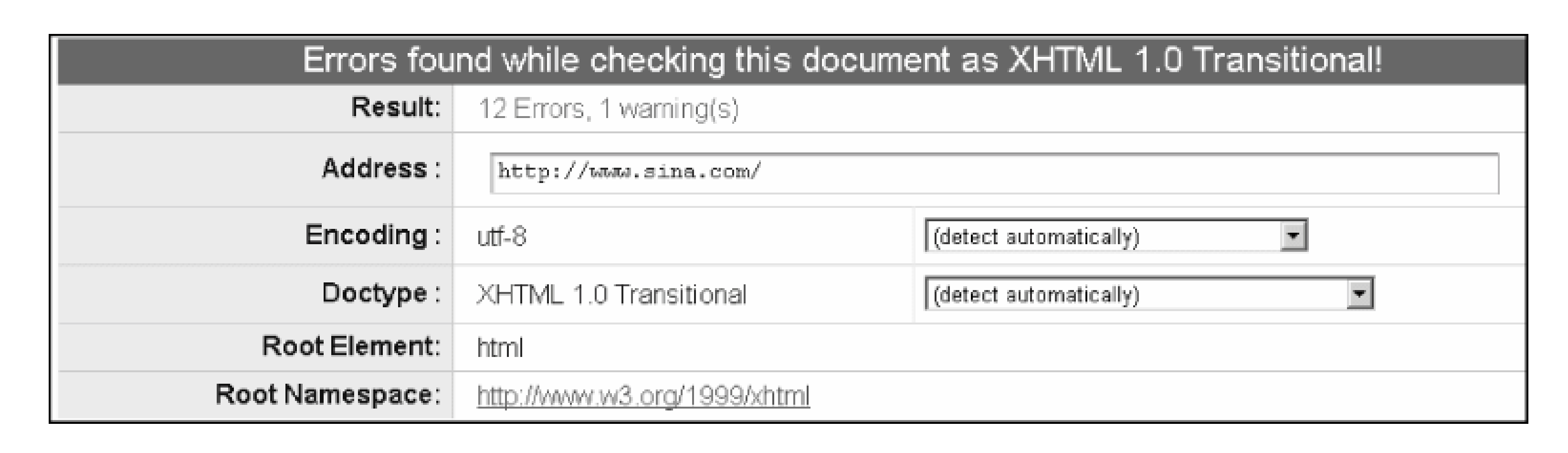


图 1.25　W3C 的标签有效性服务系统对某个著名门户网站检测的结果

应用 W3C 的标签有效性在线服务对 W3C 自身的官方网站进行检测后的结果如图 1.26 所示，说明 W3C 自身的官方网站是在"以身作则"起"表率"作用。



图 1.26　应用 W3C 的标签有效性服务对 W3C 的官方网站进行检测的结果

1.3.2　遵守 XHTML 规范设计 Web 页面

1. XHTML 文档的基本结构和基本语法要求

(1) XHTML 文档的总体结构要求

所有的 XHTML 文档必须要提供对应的文档类型声明(DocType Declaration)定义，并

且所有的 XHTML 标签都必须嵌套于<html>根标签中，而且在页面中还必须包含成对的<html>、<head>和<body>等标签，<title>标签必须放在成对的<head>标签之内。

例 1-6 为一个能够体现 XHTML 文档结构要求的代码示例，读者通过示例代码能够明确 XHTML 规范是一个“强制性”的规范。

例 1-6 体现 XHTML 文档结构要求的代码示例

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head><meta http-equiv="Content-Type" content="text/html;charset=gb2312"/>
      <title>体现 XHTML 文档结构要求的示例页面标题菜单栏中的信息文字</title>
    </head>
    <body><div id="pageHead"></div></body>
</html>
```

在例 1-6 的 XHTML 示例代码中，首先是关于文档类型文件的声明项目，然后是页面文档的开始标签<html>，但在其中声明了一个命名空间的属性(黑体所标识的代码，本示例为默认命名空间，而且是 W3C 推荐的命名空间字符串)，因为在 XHTML 文档中<html>标签内的 xmlns 属性是必需的。保证在同一个 XHTML 文档中可以包含多个不同命名空间内的标签元素，实现重用。

在成对的<head>标签内，可以提供本页面文档的设计者、页面信息的文字编码等相关内容(各种形式的<meta>标签，参见例 1-5 中的<meta>标签示例及说明文字)。

在成对的<body>标签内不同信息描述的子标签必须能够被正确地嵌套在对应的父标签体内；所有的非空标签必须能够提供对应的结束标签，而空标签如<hr>、
 和<img>等应该被替换为<hr/>、
和<img/>等形式。

(2) XHTML 文档中标签名和属性名定义的语法要求

- 依据 XHTML 规范定义，各个标签名及属性名都必须采用小写字母，并且标签名和属性名对字母的大小写敏感和严格区分。
- 所有的 XHTML 标签元素必须被嵌套于成对的<html>根标签中，并且子标签必须成对和被嵌套在其父标签中。
- 用 id 属性代替 name 属性以定义该标签的对象名、属性值包含在双括号内。

2. 常规的 HTML 页面和 XHTML 页面的 DTD 差别

DTD 定义了 HTML 及 XHTML 文档的逻辑结构，并规定了页面文档中所使用的标签、实体、标签的属性、标签和实体之间的关系等方面的信息内容。通过 DTD 文件可以检测 HTML 及 XHTML 文档的结构是否正确，DTD 是一种保证 HTML 及 XHTML 文档格式正确的有效方法。但常规的 HTML 页面的 DTD 与 XHTML 页面的 DTD 存在差别。

(1) HTML 风格页面的 DTD 定义

在 HTML 4.01 规范中提供了 3 种不同形式的 DTD 定义，其中的“HTML 4.01 Strict”严格类型的 DTD 要求在页面文档中不能包含被反对使用的各种标签和属性；而在“HTML

4.01 Transitional"过渡类型的 DTD 中,允许在页面文档中包含被反对使用的标签和属性;"HTML 4.01 Frameset"框架式类型的 DTD 则主要适用于框架集的 Web 页面定义。

下面为一个过渡类型(HTML 4.01 Transitional)的 HTML 4.01 规范中 DTD 定义的代码片段示例。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
```

(2) XHTML 风格页面的 DTD 定义

对于 XHTML 1.0 版本的页面 DTD 规范,W3C 同样也提供了 3 种类型的 DTD 定义,其中的"XHTML 1.0 Strict"严格类型的 DTD 要求在页面文档中不能包含被反对使用的各种标签和属性;而"XHTML 1.0 Transitional"过渡类型的 DTD 中,允许在页面文档中包含被反对使用的标签和属性;"XHTML 1.0 Frameset"框架式类型的 DTD 同样也主要适用于框架集类型的 Web 页面定义。

下面为一个严格类型(XHTML 1.0 Strict)的 XHTML1.0 版本规范中 DTD 定义的代码片段示例。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

3. 在 Dreamweaver 工具程序中创建符合 XHTML 规范要求的页面

(1) XHTML 规范中强制要求每个 Web 页面必须提供相应的 DTD 定义

由于在 XHTML 规范中强制要求每个 Web 页面都必须提供相应的 DTD 定义,但 3 种不同形式的 DTD 定义的代码不便于 Web 开发人员手写。因此,为了减少 Web 页面设计中的错误,在开发中可以应用各种 Web 页面设计的开发工具辅助页面设计。

(2) 应用 Dreamweaver 工具程序创建出符合 XHTML 规范要求的 Web 页面

选择 Dreamweaver 程序中"文件"主菜单中的"新建"子菜单项,将出现"新建文档"对话框(参见图 1.13);然后选择所需要创建的 Web 页面文档类型,并在"文档类型"下拉列表框中选择某种形式的 DTD 选项,如图 1.27 所示;最后单击"创建"按钮,创建出该 Web 页面,参见例 1-6 所示的代码示例。

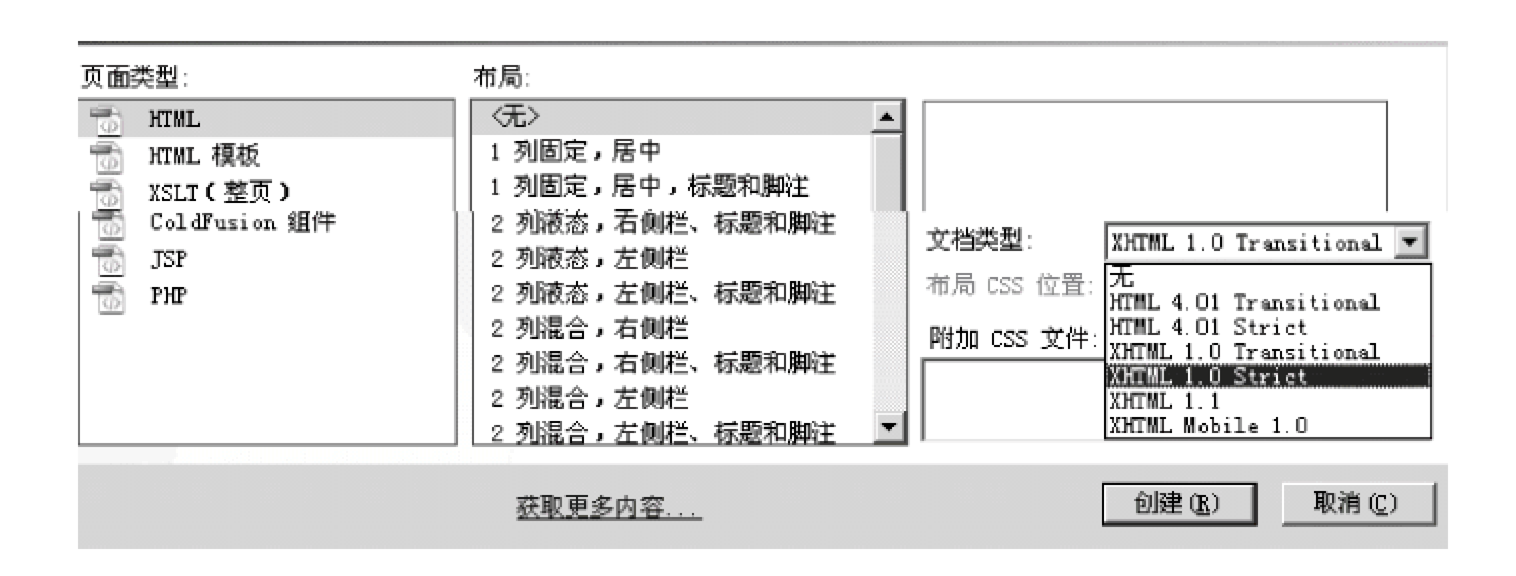


图 1.27 在 Dreamweaver 中创建符合 XHTML 规范要求的页面示例图

当 Web 开发人员设计完成该页面后,可以借助于 Dreamweaver 工具程序中所提供的基于 XHTML 页面的有效性检查功能对本页面进行语法正确性检查,及时发现不规范的标

签(参见图1.24)。

1.4 DHTML语言及应用

1.4.1 Web 2.0中的主要代表技术

1. 什么是Web 2.0

Web 2.0是一个相对的概念,早期的Web系统中只提供静态的HTML页面信息而不能进行人机交互,这被称作Web 1.0时代。随着Internet网络技术的不断发展,出现了许多动态的Web开发技术(如CGI、ASP、PHP、JSP等),并将这些技术广泛地应用在Web应用系统中,从而产生出所谓的“动态网站”,能够实现人机交互,这被称作Web 1.5时代,也就是互联网最为热门的“.com时代”。但这个时代中的Internet用户通过浏览器只能被动地获取信息,而不能直接成为信息的制造者。

在2000年,互联网及派生出的商业应用和服务开始步入“严冬萧条时节”。此时,许多互联网有志人士开始思考如何利用已有的Web技术来改造互联网及基于其上的各种应用,为用户提供更为优异的服务。

因此,在当时就提出了许多基于Web的各种服务,如Blog、Wiki、SNS、Tag等读者目前比较熟悉的各种Web服务形式,随后Web 2.0中的主要代表技术如RSS、TrackBack、Ajax等技术也逐步被提出和被广泛地应用在Web应用系统中,Web 2.0的概念也开始被明确并被深入地应用和推广。

2. Web 2.X和Web 1.X在应用方面的主要不同点

Web 1.X时代的Web应用及服务主要是以“数据”为核心,因为Web网站中所提供给用户的内容是网站编辑人员编辑加工处理后的结果,用户被动地浏览Web网站中所提供的各种信息内容,访问者不能直接参与到信息的处理(如发布、编辑和完善等)工作中。

而Web 2.X时代则改变了原来的“Web网站到用户的单向行为”,加强了Web网站与用户之间的互动,Web网站中的各种信息内容都可以基于用户直接提供,Web网站的功能建设和完善也由用户参与,最终实现了“Web网站与用户双向交流与参与”的目标。因此,Web 2.X时代的Internet是以“人为出发点”的互联网络。

3. Web 2.X中的主要代表技术

(1) 博客(Blog或Weblog,中文称为“网志”或“部落格”,或者简称为“博客”)

Blog其实是Internet上的个人共享空间,用户可以以日记的形式在网络上发表自己的个人信息内容,可以涉及从有关企业、个人、构想的新闻到日记、照片、诗歌、散文,甚至科幻小说的发表或张贴都是允许的(逐步发展成为现在比较流行的网络文学)。Weblog其实是Web Log的缩写,中文意思是“网络日志”,后来缩写为Blog;而Blogger(博客)则是写Blog

文章的人。

(2) 简单的“辛迪加”组织(Really Simple Syndication,RSS)

RSS的含义就是“内容提供商”,将各种信息用RSS格式打包(数据都是标准的XML格式,所以也能在其他的终端设备中被使用),并借助于Internet技术“推(Push)”送到用户端的本地阅读器软件中,从而可以使得用户获得所感兴趣的各种个性化信息。

RSS为信息迅速传播搭建了一个技术平台,并使得每个人都成为Internet信息的提供者。很多门户网站都提供了RSS信息频道的订阅服务,因为RSS技术具有时效性、内容全面性以及个性化定制等特点,而门户网站的业务核心也就是要给用户提供及时、丰富和有价值的信息。

(3) 播客(Podcasts或Podcasting)

“播客”这一概念来源于苹果电脑的“IPod”与“广播”(Broadcast)的合成词,它是一种在互联网上发布文件并允许用户订阅和能够自动接收新文件的方法。播客与博客(Blog)是同义词,都是个人通过互联网发布信息的方式,并且都需要借助于博客/播客发布程序进行信息的发布和管理。

但博客与播客存在一定的差别,博客所传播的信息主要是以文字和图片信息为主要形式,而播客传递的则是音频和视频信息。因此,播客其实就是一个以互联网平台为载体的个人电台和电视台。

(4) 社会性网络服务(Social Networking Services,SNS)

所谓的社会性网络服务(也称为社交网络,Social Network Site)其实就是“社交关系”的网络化,将人们现实生活中的社会交往搬到网络上,通过朋友认识朋友的形式建立属于自己的“社交圈”。

社交网络是依据六度分割理论(由美国著名社会心理学家米尔格兰姆(Stanley Milgram)于20世纪60年代最先提出)建立的网站,帮助网络用户运营朋友圈的朋友。由于在这个“社交圈”中各种人员都是自己的朋友的朋友,因此诚信和安全程度相对比较高。

(5) 多人协作的写作工具(Wiki)

Wiki的英文全称为WikiWiki,该词来源于夏威夷语的“weekeeweekee”(快点快点的意思),Wiki最适合做百科全书、知识库和整理某一个领域的知识等知识型站点。

分散在不同地区的人们利用Wiki工具能够协同工作,并共同完成对某种知识的传播。因此,Wiki技术已经被较好地应用在百科全书、手册、某个技术的FAQ(你问我答)编写、专题知识库等方面。

(6) 异步JavaScript和XML技术(Ajax)

Ajax(Asynchronous JavaScript And XML),中文翻译为“异步JavaScript和XML技术”。Ajax技术可以使Web应用具有类似于桌面客户端软件的交互效果,从而给基于Web的应用系统在加载动态化和操作客户端化两方面带来了革命性的突破。

不需要刷新整个页面或者局部刷新就可以实现向服务器端程序提交Http请求信息,因为Ajax技术能够提供让客户端程序与服务器端程序进行异步通信的能力,从而使用户从请求/响应中解脱出来,最终提高了Web应用系统的响应效率。

但 Ajax 本身并不是一门新的语言或技术，它实际上是几种技术按一定的方式组合、协作的“复合”技术，其中主要包含 XMLHttpRequest、JavaScript、XML、XHTML、DHTML 和 CSS 等核心技术。下面重点介绍 DHTML 标签语言及其在客户端的应用。

1.4.2 DHTML 标签语言及应用

1. 什么是 DHTML 标签语言

(1) DHTML 不是由 W3C 提出的技术标准

所谓 DHTML(Dynamic HTML，动态 HTML)并不是一门新的标签语言，更不是 W3C 规定的标准，它实际上是将 HTML(XHTML)、HTML 层(Layers)、CSS、DOM(Document Object Model，文档对象模型)和 CSSL(Client-Side Scripting Language，客户端脚本语言，如 JavaScript、VBScript、JScript 等)等集成在一起的技术。

因此，DHTML 不是一种标准或规范，而是一种组合技术。DHTML 不是由 W3C 提出的技术标准，而是由浏览器厂商(主要是网景公司和微软公司)根据自己的需要在各自的浏览器中支持的整合技术。不同的浏览器对 DHTML 及相关的技术支持是不同的，这也为应用 DHTML 技术带来了一定的不便和不兼容性。

(2) 应用 DHTML 技术能够增强 Web 应用系统客户端的人机交互性

应用 DHTML 技术，其实就是以 HTML(XHTML)为基础，运用 HTML DOM 将页面中的各个元素转换为 DOM 树模型中的节点对象，并利用 CSSL(客户端脚本语言)控制这些节点对象的 CSS 属性，以达到动态改变 Web 网页中的视觉显示效果。

DHTML 技术是目前比较成熟的 Web 客户端应用技术，应用 DHTML 技术可以实现在浏览器客户端直接动态地更新 Web 网页中的信息内容、排版样式、播放动画等而无须经过服务器端程序的处理。比如，将鼠标移至某个超链接时将自动地显示相关的信息或者改变超链接的外观；而单击该超级链接后会自动显示出下拉式的子项目等。

2. DHTML 和标准的 HTML 之间的关系

DHTML 建立在原有 Web 客户端技术之上，但进一步改进了动态交互性，使得静态的 HTML 变成了可动态操控的 DHTML。两者之间的关系及在应用方面的差别主要体现在以下几个方面。

(1) 动态改变页面中的内容(Dynamic Content)

通过对 HTML(XHTML)页面中的各种标签唯一地标识对象名(设置标签的 id 属性值)，Web 开发人员就可以在客户端的脚本程序中动态地更新网页中的内容，如插入、修改或删除信息内容等操作行为。

(2) 动态改变样式表(Dynamic Styles Sheets)

CSS 中的各种属性本来也是静态的，但在 DHTML 技术中通过应用客户端的脚本程序也可以被动态操控，从而可以产生出动态的显示效果——动态地改变样式表。比如：隐藏或显示内容、修改样式定义、激活标签实体等方面的功能。

(3) 动态定位(Dynamic Positioning)

由于在HTML(XHTML)规范中提供了层(Layers)的概念,HTML标签实体具有X、Y、Z轴三个不同方向的定位,并允许在客户端的脚本程序中动态地改变三个方向的坐标和定位页面中的任何对象,而调整不同实体在Z轴方向上的层叠次序,就可以产生重叠显示的效果。

3. 为什么各个浏览器厂商要提供并推广DHTML技术

应用DHTML技术可以使得Web开发人员创建出能够与用户交互并包含动态内容的页面,以进一步增强Web应用系统的人机交互性,但又不依赖于某个特定的服务器端技术实现的平台。不同浏览器厂商的DHTML技术现在已经逐步被"标准化"到W3C的HTML/XML DOM规范中,同时也推动了目前比较主流的Ajax技术的应用普及,因为Ajax技术其实是DHTML技术思想的进一步完善,扩展到可以与服务器端程序进行交互。

本章小结

教学重点

HTML标签语言是Web应用系统客户端开发实现的基础性语言,但W3C颁布的HTML规范中存在数量众多的标签,在教学中一方面没有足够的课时全面讲授所有的HTML标签,另一方面也没有必要。因为标签语言与一般的编程语言一样,也是有"共性"的。因此,在教学中一定要把握HTML标签的"共性",可以按照分类挑选典型的标签作为示例讲解,这样的教学方式能够使得学生快速掌握HTML标签语法规则和应用HTML标签设计页面。

对于与XHTML内容有关的教学,重点在于说明Web标准和规范的重要性,同时解释W3C为什么要抛弃HTML而推荐XHTML,以及XHTML和HTML有什么本质上的差别,为后续进一步学习XML强化规范意识。

而对于DHTML方面教学内容的重点在于解释清楚DHTML到底是什么,各个浏览器厂商为什么要提出DHTML技术,应用DHTML技术可能带来的兼容性问题等,但对于具体如何应用DHTML技术,在后续的章节中进一步学习时再讲解。我们目前处在Web 2.0时代,应该对Web 2.0中的主要代表技术有选择性地进行介绍,以扩展学生的知识面,并激发学生对Web开发技术学习的兴趣。"知之者不如好之者,好之者不如乐之者"。

学习难点

遵守Web标准的核心思想是要求Web开发人员采用合适的HTML标签设计和实现页面。比如,要描述页面文字中的一个段落,就应该采用<p>标签而不要应用<div>标签;导航菜单栏中各个项目的定位应该采用<ul>等标签,而不应该采用表格中的<td>标

签；对于页面内容的显示实现技术，则应该采用CSS来控制，而不应该直接应用标签中的样式属性项目定义显示风格。

因此，目前Web标准的开发方法为什么要推荐使用“XHTML＋CSS＋Div＋JS”开发和制作Web网站，其主要目的是使“结构（内容）”与“表现”彻底分离。其中的XHTML标签只用来定义页面文档的“结构”或者“内容”，所有涉及如何“表现”的定义项目都应该放到一个独立的CSS样式定义文件中。

遵守Web标准、增强应用Web标准的意识和明确为什么要满足Web标准的要求，这不仅是作为学生的读者学习本章的一个主要难点，也是作为教师的读者在教学中的难点。

教学要点

HTML文档主要分为非框架集页面文档和框架集页面文档两种类型，在教学过程中需要通过具体的示例讲解清楚这两种不同的文档在语法规则和结构方面的差别。

在Web标准的开发方法中推荐采用<div>（或<span>）标签实现页面内容的布局定位，而不再应用<table>标签；在讲解HTML表格<table>标签时，不应该仅仅停留在普通的表格设计和实现方面，而应该通过示例说明如何创建跨多行、多列的表格，如何创建行分组或列分组的表格。同样在讲解HTML表单<form>标签时，也不应该仅仅停留在普通的表单设计和实现方面，还应该通过示例说明如何应用<fieldset>标签实现对Web表单中的项目分组。

XHTML是更规范的HTML，W3C发布XHTML规范的主要目的是希望Web开发人员能够设计并实现比较“严谨”和“规范”的HTML页面。因此，在教学中也应该强调“规范”和“标准”的重要性，并要求学生在页面设计方面的练习或作品应该用XHTML实现。

学习要点

读者在学习本章的内容时应该不会有学习方面的难点，因为一方面，在本章中没有出现抽象的名词和术语；另一方面，HTML标签语言中的各个标签本身具有自描述特性，各个标签简洁、易理解。但由于HTML标签数量众多，每个标签又有许多不同的属性项目，因此在学习时会有比较“繁杂”的感觉。

建议读者在学习每个HTML标签时，首先将HTML标签按照功能分类，进行分类学习；其次在学习每个标签时都编写具体的页面示例并在浏览器中预览，直观地理解每个标签的功能，因为“理性构建在感性基础之上”。这样的学习方法能够达到“事半功倍”的学习效果。

本章练习

1. 单选题

(1) 下面各项为<input>标签的type属性的值，哪一项表示的是表单中的提交按钮？

A. type="text" B. type="radio"
C. type="image" D. type="file"

(2) 下列哪一项能够实现将<div>标签内的文字居中对齐?

A. <div align="middle">…</div> B. <div align="right">…</div>
C. <div align="left">…</div> D. <div align="justify">…</div>

(3) 下面哪一个标签能够描述和定义 Web 表单?

A. <body> B. <font> C.
 D. <form>

(4) 下列哪一项能够实现在一个新的浏览器窗口中打开某个网页文档?

A. target="_self" B. target="_blank"
C. target="_top" D. target="_parent"

(5) 在 HTML 标签语言中,<body bgcolor=?> 标签表示为下面哪一项的含义?

A. 设置背景颜色 B. 设置文本颜色
C. 设置链接颜色 D. 设置已使用的链接的颜色

2. 填空题

(1) 创建一个 HTML 文档的开始标签是________,结束标签是________。

(2) 设置 HTML 文档的标题以及其他不在 Web 页面上显示的信息的开始标签是________,结束标签是________。

(3) 设置 HTML 文档的可见部分的开始标签是________,结束标签是________。

(4) HTML 页面中的标签代码 表示的含义是________,其中 href="url"的含义是________。

(5) HTML 代码<frameset rows="value,value" border="1"> 表示的含义是________,其中 border="1"的含义是________。

3. 问答题

(1) 标签语言的主要特点是什么? 它与其他程序设计语言的区别主要体现在哪些方面?

(2) 为什么说用 HTML 标签编写的是静态网页? HTML 在应用方面存在哪些方面的不足?

(3) W3C 为什么要放弃 HTML 标签语言? 应用 XHTML 标签语言时需要遵守哪些基本的语法要求?

(4) Web 2.0 中的主要代表技术有哪些? 请描述 DHTML 和标准的 HTML 之间的关系。

(5) 什么是 Ajax 技术? 应用 Ajax 技术能够为应用系统带来哪些方面的改变?

4. 开发题

(1) 按照图 1.28 所示的表格及数据在显示方面的要求,请写出对应的 HTML 页面标签代码。

(2) 按照图 1.29 所示的表单及各个输入控件的要求,请写出对应的 HTML 页面标签代码。

姓名	性别	年龄	学校
刘三	男	21	蓝梦大学
万五	女	22	蓝梦学院
李一	男	21	蓝梦技校
章二	女	20	蓝梦技校

图 1.28　描述学生信息的表格

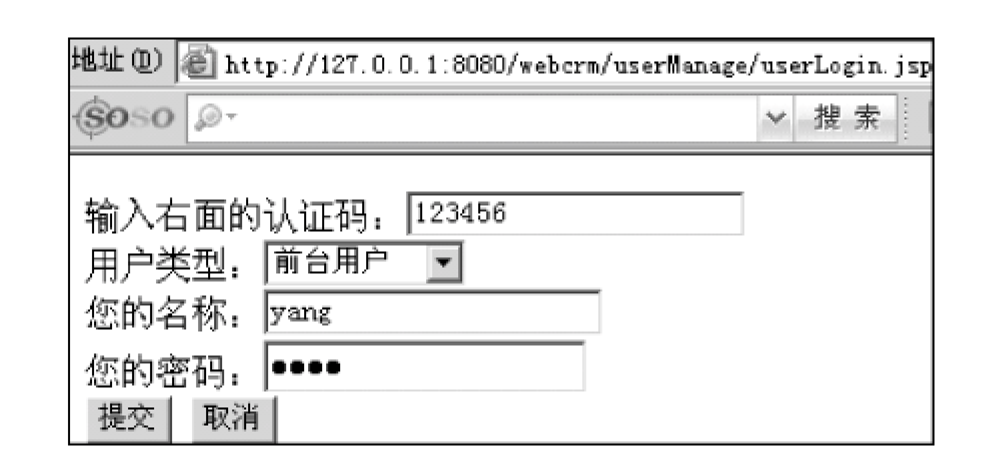


图 1.29　某个 Web 应用系统中的用户登录页面表单

(3) 按照图 1.30 所示的某个 Web 应用系统的全局导航菜单栏和其中的各个超链接的显示要求，请写出对应的各个超链接的 HTML 标签代码。



图 1.30　某个 Web 应用系统的全局导航菜单栏和其中的各个超链接

第 2 章 基于对象的 JavaScript 语言

CHAPTER 2

JavaScript 作为 Web 应用系统的客户端脚本语言其实早已经被广泛地应用，但在目前的 Web 2.0 技术中又被进一步地发展和完善，成为目前许多基于 Web 2.0 技术的 Web 应用系统中首选的脚本控制语言。JavaScript 广泛地应用在 Ajax、jQuery、Flex 等技术实现中，大大地增强了 Web 应用系统的人机交互能力，各种基于 JavaScript 语言的 RIA(Rich Internet Application，富因特网应用程序)应用系统也层出不穷。

尽管 JavaScript 脚本语言曾经被认为是“玩具级别”的语言，但是在多位 Web 开发大师的精心完善和努力推广下，它又重新获得了 Web 开发人员应有的重视；同时 JavaScript 也作为 ECMA 制定的语言标准，在 Web 表示层的开发实现技术中起着非常重要的作用。

在本章中初步学习和掌握 JavaScript 语言基础知识，主要涉及数据类型、表达式和操作符、语句及控制流，还涉及函数及具体的编程应用。面向对象编程中的对象及具体的编程应用等方面的内容。而在后续的章节中，将深入地学习和应用 JavaScript 语言。

2.1 JavaScript 语言基础

2.1.1 JavaScript 语言概述

1. 什么是 JavaScript 语言

JavaScript 是一种基于对象(Object)和事件驱动(Event Driven)的脚本语言，在 20 世纪 90 年代提出它的主要目的，是希望能够直接在 Web 客户端浏览器中产生复杂的人机交互行为，增强基于 Web 方式的应用系统的动态交互功能。

JavaScript 的出现弥补了 W3C HTML 标签语言固有的缺陷，因为 HTML 语言只能“描述”而不具有行为定义的特性。JavaScript 其实是 Sun 公司的面向对象 Java 语言与 HTML 标签语言相互折中的技术特性的结果，同时它也是目前 Ajax 技术实现的主要编程语言。

2. JavaScript 脚本语言的主要技术特性

（1）它是一种脚本语言并且采用小程序片段的方式编程

与其他脚本语言（如 SQL，结构化查询语言）一样，JavaScript 同样也是一种解释性的脚本语言。它的语法规则和程序结构元素都类似于面向对象的 Java 语言，但它又不像 Java 语言那样需要先编译后执行，而是在程序运行过程中直接被 Web 浏览器逐行地解释执行。因此 Web 浏览器是 JavaScript 脚本代码的解释器程序，而且目前各个厂商的 Web 浏览器都支持 JavaScript 脚本语言。

JavaScript 脚本代码直接内嵌到 HTML 页面中，并且可以动态地操控（如新增、修改和删除）页面中的各个 HTML 标签元素对象，从而直接在 Web 客户端浏览器中产生复杂的人机动态交互行为，使得 Web 应用系统的动态交互功能的实现不再仅仅依赖于服务器技术。

（2）它是基于对象的脚本语言

JavaScript 是一种基于对象（但不是面向对象）的脚本语言，这意味着它具有面向对象编程技术中的“抽象”和“封装”的特性，但又不具有“继承”和“多态”的特性，也就是不能直接子类型化对象；在 JavaScript 语言中已内置了若干系统级的对象，开发人员可以直接应用它们，当然也允许开发人员根据应用的需要自定义特定功能的对象。

（3）它是一种编程实现简单的脚本语言

JavaScript 脚本语言的简单性主要体现在如下两个方面：

- 首先它保留了 Java 语言的基本语法规则，如数据类型、运算符、语句和控制流等，但又对 Java 语言中的许多语言特性及语法规则进行了简化设计，这对于熟悉 Java 语言的 Web 开发人员来说，非常熟悉和能够快速应用。
- 其次它的变量类型的定义采用“弱类型”的语法规则，并未使用严格的数据类型，这在一定的程度上也简化了程序代码中各种变量的声明和使用。

（4）它在浏览器页面中各种事件的触发下被动执行

JavaScript 脚本代码是被浏览器动态加载和执行的，它可以直接对用户的各种输入做出事件响应，无须经过 Web 服务器端有关程序的功能处理。JavaScript 脚本语言所具有的事件驱动特性，也就是指用户在页面中执行了某种操作而产生某个动作后，这个动作行为就称为“事件”（Event）。比如用户按下鼠标、移动浏览器的窗口、选择某个菜单等都可以看成某个特定的事件，而一旦事件触发，就可以执行相应的事件响应函数的代码（一般为 JavaScript 程序代码）。

（5）它具有跨系统平台特性

JavaScript 脚本语言只依赖于 Web 浏览器，但与浏览器程序本身所在的机器操作系统的环境无关，只要能运行支持 JavaScript 脚本语言的 Web 浏览器就可正确地执行 JavaScript 脚本程序，从而实现了类似 Java 语言的“一次编写、到处执行”的跨系统平台效果。

3. JavaScript 脚本语言和 Java 编程语言的区别

JavaScript 脚本语言不是 Sun 公司的产品，它是由当时的 Netscape 公司为主并联合多

家公司开发的，并且是对ECMAScript(European Computer Manufacturers Association，欧洲计算机制造商协会)标准的扩展。虽然JavaScript与Java有紧密的联系，但却是两个不同公司开发的产品，并且是基于不同的设计思想而构建的语言。

尽管JavaScript和Java很类似，但本质上又不一样！Java是一种比JavaScript更复杂、功能更强大的面向对象的编程语言，而JavaScript则是对Java语言进行“精华”和“瘦身”后的结果(或者称为Java语言的简化版)。下面对两种语言的异同进行对比。

(1) 基于对象和面向对象

- Java语言是一种真正的面向对象(Object Oriented)的编程语言，因为它完整地支持面向对象编程技术中的“抽象”、“封装”、“继承”和“多态”等特性。
- JavaScript是一种基于对象(Object Based)和事件驱动的脚本类型的语言，因为它只支持面向对象编程技术中的“抽象”、“封装”特性，而没有支持“继承”和“多态”等方面的特性。因此，也就不能直接产生子类型或者子对象。

(2) 解释执行和编译执行

- Java语言的源程序代码必须首先经过编译并产生*.class的类文件指令码后，才可以由Java虚拟机解释执行，因此Java是一种“编译解释”型的语言。
- JavaScript却是一种解释性的语言，它的源程序代码直接由Web浏览器解释执行，而不需要进行任何的预先编译。

(3) 强变量定义和弱变量定义

- Java语言采用“强类型”变量的语法检查机制，即程序中的所有变量在使用之前都必须声明和正确地赋值，而且所赋值的数据类型和定义的类型要保持一致性。如下示例中的x为一个数值型变量，而变量y为字符串型变量。

```
int x = 1234;
String y = "4321";
```

- JavaScript中的变量声明则采用“弱类型”变量的语法检查机制，即变量在使用前可以不进行预先的声明，而是由Web浏览器在执行对应的代码时根据变量的具体值推断出变量的实际数据类型。在如下示例代码中的变量x为一个数值型变量，而变量y为字符串型变量。

```
x = 1234;
y = "4321";
```

4. 在HTML页面中如何应用JavaScript脚本程序代码

在HTML文档中加入JavaScript脚本代码有两种不同的方法：内部嵌入式和外部链接式。内部嵌入式就是把JavaScript代码直接嵌入在HTML文档中；外部链接式则是把JavaScript代码放置在一个外部的独立文本文件中，该文件的扩展名一般定义为*.js，然后在HTML文档中引入该外部*.js文件中的JavaScript脚本代码。

(1) 直接内嵌到HTML页面的<head>标签区域中

JavaScript程序中各个基本的组成成分，如变量定义、函数定义和调用等方面的程序代

```
<xsl:template name = "someOneSubTemplate">
     <! -- 模板规则中的内容定义 -->
</xsl:template>
<xsl:template match = "someOneTemplate">
    <xsl:call - template name = "someOneSubTemplate"/>
     <! -- 模板中的其他内容定义  -->
</xsl:template>
```

属性 priority 也是可选属性，用于指定模板的优先级。当出现模板冲突时，决定哪个模板将被首先调用，优先级数值最大的模板会被首先执行；属性 mode 也是可选属性，用于指定模板的样式。当多个不同的模板作用于同一个 XML 节点时，可以通过 mode 属性区分不同的模板，而在调用时使用 mode 属性标识模板。

4. 什么是匹配模式

所谓的匹配模式是用来定位 XML 源文档中哪个节点或哪些节点集将被 XSLT 模板处理的条件集，只有满足匹配条件的 XML 节点元素才会被 XSLT 模板处理，未匹配的其余 XML 节点将被忽略。因此，能够按照指定的逻辑关系或数据访问的条件操作 XML 文档中的数据，XSLT 样式表也提供了与编程语言类似的逻辑控制能力。

但<xsl:template>标签元素只能作为<xsl:stylesheet>标签元素的子元素出现，而且<xsl:template>标签元素之间不能相互嵌套。

5. 模板规则所期望输出的内容

<xsl:template>标签元素体内的内容为模板规则的输出内容，由一系列的其他 XSLT 指令元素所构成，如例 7-5 示例所示。模板规则的内容决定了 XSLT 最终输出的形式和结果值。比如，如果希望在 Web 浏览器中输出，则模板规则的内容应该包含 HTML 标签文本。当然，如果希望在手机等掌上设备中输出，则模板规则的内容应该包含 WML 标签文本。

因此，替换不同的模板规则的内容，就可以实现对同一个 XML 文档以不同形式输出。

6. <xsl:template>标签元素的应用示例

例 7-8 为某个描述课程信息的 XML 文档示例，黑体所标识的标签指定目标 XSLT 文件。下面的模板定义语句匹配例 7-8 示例中所有的<oneCourseInfo>标签元素内的内容。

```
<xsl:template match = "oneCourseInfo"></xsl:template>
```

下面的模板定义语句匹配例 7-8 示例中的<oneCourseInfo>标签元素子节点中的<content>标签元素，因为其中的路径指示符“/”表示直接的父子节点关系(为分隔符)。

```
<xsl:template match = "(oneCourseInfo/content)"></xsl:template>
```

下面的模板定义语句匹配例 7-8 示例中的父节点为<oneCourseInfo>标签元素的后继节点中所有的<content>标签元素，因为<content>标签元素可能会重复出现。

```
<xsl:template match="oneCourseInfo//content"></xsl:template>
```

下面的模板定义语句匹配例 7-8 示例中的 XML 文档中的根节点。

```
<xsl:template match="/"></xsl:template>
```

例 7-8 描述课程信息的 XML 文档示例

```
<?xml version="1.0" encoding="GB2312" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="courseInfos.xsl"?>
<courseInfos>
  <oneCourseInfo>
     <title>XML 技术与应用的课程介绍</title>
     <teacher>张老师</teacher>
     <startDate>2010 年 9 月 1 日</startDate>
     <content>XML 是一种标签语言而且是用来创造标签语言(比如 HTML)的元语言……</content>
  </oneCourseInfo>
</courseInfos>
```

7.2.2 <xsl:apply-templates> 标签的语法及应用

1. <xsl:apply-templates>标签元素的语法

此标签元素一般和<xsl:templates>标签元素相互结合使用，通过<xsl:templates>标签确定模板规则，以及哪个 XML 节点元素将被匹配，然后利用<xsl:apply-templates>标签元素在当前位置(在此位置处将插入满足匹配条件的模板规则所定义的内容)激活在此位置以下所匹配的第一个模板，如例 7-10 示例中对<xsl:apply-templates>标签元素的应用示例。

应用模板规则的<xsl:apply-templates>标签元素的语法定义示例如下所示：

```
<xsl:apply-templates select ="节点表达式" />
```

示例中的 select 属性定义 XML 文档中的节点名称，如果在<xsl:apply-templates>标签中没有应用 select 属性，则表示需要处理 XML 文档中所有当前节点的子标签元素；如果应用了 select 属性，则处理的目标 XML 标签元素将由 select 属性的表达式值决定。此时标签元素<xsl:apply-templates>仅会处理与 select 属性值相匹配的 XML 子标签元素。

2. 应用<xsl:apply-templates>标签元素时要注意的问题

<xsl:apply-templates>标签元素可把一个模板规则应用于 XML 文档中当前的标签元素或者当前标签元素的子标签节点上，但它总是要包含在<xsl:templates>标签元素内，属于它的子标签。下面的模板规则定义语句匹配例 7-8 示例中整个 XML 文档，也就是从 XML 文档根节点(<courseInfos>标签的父节点)开始匹配，但具体执行时只处理根节点下所有的<oneCourseInfo>标签元素节点内的数据。

```
<xsl:template match="/">
```

```
        <! -- 其他的 HTML 标签在此省略 -->
        <xsl:apply-templates select="oneCourseInfo"/>
        <! -- 其他的 HTML 标签在此省略 -->
</xsl:template>
```

下面的模板规则定义语句匹配例 7-8 示例中的<oneCourseInfo>节点，所有<oneCourseInfo>节点下的子标签元素节点都将被模板处理，因为未给定 select 属性定义。

```
<xsl:template match="oneCourseInfo">
        <! -- 其他的 HTML 标签在此省略 -->
        <xsl:apply-templates />
        <! -- 其他的 HTML 标签在此省略 -->
</xsl:template>
```

3. XSLT 中的模板规则定义和激活的应用示例

在如例 7-8 所示的描述课程信息的 XML 文档示例中的 XSLT 样式文件 courseInfos.xsl 的内容如例 7-9 所示，在该 XSLT 样式文件中首先利用<xsl:templates>标签元素定义一个从 XML 根节点开始匹配的模板规则(黑体所标识的标签)，然后在该模板规则定义中应用<xsl:apply-templates>标签元素(黑体所标识的标签)激活下一个模板规则定义，并最终获得 XML 文档中目标标签节点中的数据。

例 7-9 体现 XSLT 中的模板规则定义和激活的代码示例

```
<?xml version="1.0" encoding="gb2312" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl" version="1.0">
    <xsl:template match="/">
        <html><head><title>对课程信息的 XML 文档应用 XSLT 样式示例</title></head>
            <body><xsl:apply-templates select ="courseInfos"/></body>
        </html>
    </xsl:template>
    <xsl:template match="courseInfos">          定义一个匹配的目标起点为
                                                courseInfos 节点的模板规则
        <xsl:apply-templates select ="oneCourseInfo"/>
    </xsl:template>
    <xsl:template match="oneCourseInfo">        定义一个匹配的目标起点为
                                                oneCourseInfo 节点的模板规则
        <xsl:value-of select="title" />: 授课教师由本校优秀教师
        <b><xsl:value-of select="teacher"/></b>全程授课,该课程的开课时间是:
        <b><i><xsl:value-of select="startDate"/></i></b>,课程内容的简要说明如下:
        <b><xsl:value-of select="content" /></b>
    </xsl:template>
</xsl:stylesheet>                               获取满足匹配要求的
                                                指定标签中的数据
```

例 7-9 也反映了 XSLT 模板规则之间可以相互串接和组装的技术特性，可以操作 XML 文档中各个层次的标签节点。如例 7-8 所示的描述课程信息的 XML 文档示例最终在浏览器中预览的结果如图 7.7 所示，在例 7-9 所示的 XSLT 样式示例中应用了 HTML 中的粗体<b>和斜体<i>标签对输出的内容进行格式化控制，从而产生特定格式的排版效果。

```
            <html>
               <head>
                  <title>北京蓝梦大学软件学院学生信息</title>
               </head>
               <body><xsl:apply-templates select = "软件学院学生信息"/></body>
            </html>
   </xsl:template>
   <xsl:template match="软件学院学生信息">
      <h3>下面为北京蓝梦大学软件学院学生信息表</h3>
      <table border="1">
         <th>姓名</th><th>性别</th><th>出生日期</th>
         <th>专业</th><th>班级</th><th>专业方向</th>
         <xsl:apply-templates select = "学生信息" />
      </table>
   </xsl:template>
   <xsl:template match="学生信息">
      <tr>
          <td><xsl:value-of select="姓名"/></td>
          <td><xsl:value-of select="@性别"/></td>
          <td><xsl:value-of select="出生日期"/></td>
          <td><xsl:value-of select="专业"/></td>
          <td><xsl:value-of select="班级"/></td>
         <xsl:apply-templates select = "班级"/>
      </tr>
   </xsl:template>
   <xsl:template match="班级">
        <td><xsl:value-of select="@方向"/></td>
      </xsl:template>
   </xsl:stylesheet>
```

应用下一个模板规则，也就是下面的"软件学院学生信息"模板

应用下一个模板规则，也就是下面的"学生信息"模板

显示学生信息，select 为匹配模式，其中"@性别"代表"学生信息"标签中的"性别"属性

匹配 XML 文档中的"班级"标签，其中"@方向"获得"班级"标签中的"方向"属性

例 7-10 示例中的 studentInfo.xml 文档在浏览器中预览的结果如图 7.8 所示，XML 文档中与学生有关的各个方面的信息在表格中显示输出。



姓名	性别	出生日期	专业	班级	专业方向
张三	男	1987/10/18	软件工程	软件1班	J2EE软件开发
李四	女	1988/2/18	软件工程	软件2班	数字媒体方向

图 7.8　例 7-10 示例中的 XML 文档在浏览器中预览的结果

3. 为同一个 XML 文档提供不同显示风格的 XSLT 样式文档

例 7-12 示例是对例 7-10 示例中 studentInfo.xml 文档提供的另一个 XSLT 样式示例，在该示例中对 XML 文档中的数据以另一种风格显示。

例 7-13 一个描述电影信息的 XML 文档内容示例

```
<?xml version = "1.0" encoding = "gb2312"?>
<?xml - stylesheet type = "text/xsl" href = "showFilmByHTML.xsl"?>
<电影片库>
   <电影 id = "1">
      <片名>西游记</片名>
      <类型>电影</类型>
      <时代>古典</时代>
   </电影>
   <电影 id = "2">
      <片名>大决战</片名>
      <类型>电影</类型>
      <时代>现代</时代>
   </电影>
   <电影 id = "3">
      <片名>红楼梦</片名>
      <类型>电视连续剧</类型>
      <时代>现代</时代>
   </电影>
</电影片库>
```

为例 7-13 示例中的 XML 文档内容设计的以 HTML 表格形式输出的 XSLT 文档如例 7-14 所示，该 XSLT 文档的文件名为“showFilmByHTML.xsl”。其中黑体所标识的标签为引用外部 CSS 样式文件“filmStyle.css”。

例 7-14 以 HTML 表格形式输出的 XSLT 文档示例

```
<?xml version = "1.0" encoding = "gb2312"?>
<xsl:stylesheet version = "1.0"
                xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
   <xsl:output method = "html" version = "4.0"/>
   <xsl:template match = "/">
      <html><head><title>流行电影大片</title>
         <link rel = "stylesheet" type = "text/css" href = "filmStyle.css"/>
         </head><body><table border = "1" width = "200">
               <tr><th>片名</th><th>类型</th><th>时代</th></tr>
               <xsl:for - each select = "/电影片库/电影">
                  <xsl:sort select = "Format"/>
                  <tr><td><xsl:value - of select = "片名"/></td>
                        <td><xsl:value - of select = "类型"/></td>
                        <td><xsl:value - of select = "时代"/></td>
                  </tr>
               </xsl:for - each>
            </table></body></html>
   </xsl:template>
</xsl:stylesheet>
```

例 7-15 中的代码为例 7-14 中的 HTML 表格所需要的 CSS 样式定义示例，为了节省本

(3) <xsl:strip-space>

XSLT 在处理 XML 文档文本之前需要将其中的空白符如＃x20(空格)、＃x9(Tab)、＃xD(回车)或＃xA(换行)等剥离(Strip)掉,<xsl:strip-space>标签元素定义应当删除哪些元素中的空白字符。如示例:<xsl:strip-space elements="姓名 班级" />,表示删除<姓名>和<班级>元素中的空白字符。

(4) <xsl:preserve-space>

定义需要保留空白字符的元素。由于保留空白字符为 XSLT 中的默认设置,因此,一旦使用了<xsl:strip-space>元素,也就必须要使用<xsl:preserve-space>元素指定 XML 文档中的哪些元素需要保留空白字符。如示例:<xsl:preserve-space elements="出生日期 专业" />,表示需要保留<出生日期>和<专业>元素中的空白字符。

(5) <xsl:output>

定义输出文档的类型和格式(XML、HTML、TEXT 等),其中的 method 属性为可选属性,用于指定输出文档的格式。当然,如果未指定该属性,默认输出为 XML 格式;另外,如果在输出的目标文档中的根元素是<html>,将输出 HTML 格式的文档。

method="text"的输出内容中只有节点的文本,而 method="xml"的输出结果中自动创建并输出 XML 文档声明指令,但 method="html"时将不输出 XML 文档声明指令。

(6) <xsl:key>

声明一个命名的“键-值”对(哈希表的字典对象),使得在利用 XPath 中的 key()函数时可以更方便地操作复杂的 XML 文档。其中的 name 属性规定“键-值”对的名称,match 属性定义该“键-值”对被具体应用到哪个节点,而 use 属性指定要作为该“键-值”对的值使用的表达式。

如示例:<xsl:key name="学生信息哈希表" match="学生信息" use="@性别"/>,为 XML 文档中的<学生信息>,节点创建了一个命名的字典对象,以“性别”属性作为键(Key),生成一个数据集。在应用时,可以采用指定的名称引用该字典对象:<xsl:for-each select="key('学生信息哈西表', '女')">,最终在原来生成的数据集中按照定义的键(本示例为“性别”)检索目标节点的数据,提高数据检索的效率。

(7) <xsl:decimal-format>

定义在应用 format-number()函数把数字转换为字符串时,所要使用的分隔小数与整数的字符,因为不同国家的小数点(中文采用“.”)和数字分组分隔符(中文采用“,”)是不同的。其中的 name 属性定义一个名称,而 decimal-separator 属性规定小数点的字符(默认是“.”),grouping-separator 属性规定千分位的分隔字符(默认是“,”)。

如示例:<xsl:decimal-format decimal-separator="." grouping-separator="," name="oneDigitChar" />,定义了一个名称为“oneDigitChar”的格式,然后在下面的示例中引用(由 format-number()函数中的第三个参数决定)该名称的格式定义:<xsl:value-of select="format-number(123456.789,'#.###,00','oneDigitChar')"/>。

(8) <xsl:namespace-alias>

把样式表中的命名空间替换为其他形式的命名空间字符串,也就是使用其他的前缀符替换与给定命名空间关联的前缀符,从而形成当前命名空间的别名。因为试图在一个 XSLT

```
<xsl:variable name="XML课程平均成绩">
    <xsl:value-of select="sum(.//学生信息/XML考试成绩) div count(学生信息)"/>
</xsl:variable>
```

(11) <xsl:param>

声明一个局部或全局参数，它类似于<xsl:variable>指令元素，并且有与<xsl:variable>相同的属性和相同的功能定义。但<xsl:param>指令声明的是"参数"，在调用某个模板时可以将<xsl:param>声明的参数传递到模板中；而<xsl:variable>指令声明的是"符号常量"，为其他的 XSLT 指令提供命名的符号值。

<xsl:param>指令元素常和<xsl:with-param>指令元素配合使用，将定义的参数传递到一个模板中。但<xsl:with-param>元素中的 name 属性的值必须要与<xsl:param>元素中的 name 属性的值相匹配，否则将忽略<xsl:with-param>元素的作用。如下代码片段定义了一个名称为"loopTemplateName"的模板，实现循环功能，并定义了一个名称为"loopCount"的参数作为循环的计数器。

```
<xsl:template name="loopTemplateName">——指定模板的名称
    <xsl:param name="loopCount"/>
    <xsl:if test="$loopCount > 0">
        <xsl:text>执行循环体语句中的功能</xsl:text>
        <xsl:call-template name="loopTemplateName">——递归调用模板
            <xsl:with-param name="loopCount" select="$loopCount - 1"/>
        </xsl:call-template>                                循环计数减1
    </xsl:if>
</xsl:template>
```

只有应用了<xsl:with-param>标签元素后，参数变量才真正地成为"可变化"的变量，可以根据当前上下文的需要重新为参数变量赋值。但用<xsl:with-param>标签元素所赋的变量值，只在调用的模板中有效。而下面的代码片段示例调用名称为"loopTemplateName"的模板，并为模板中的命名参数 loopCount 赋值为 10(代表实际将要执行 10 次循环)。

```
<xsl:template match="/">
    <xsl:call-template name="loopTemplateName">——调用指定名称的模板
        <xsl:with-param name="loopCount">10</xsl:with-param>
    </xsl:call-template>
</xsl:template>                                 定义实际的循环次数
```

由于<xsl:param>元素与<xsl:variable>元素都可以声明指定名称的变量，但在同一个上下文环境中应用时，所声明的变量不能同名，否则将为重复声明。

2. 定义模板规则的 XSLT 标签元素

<xsl:template>标签元素是应用 XSLT 技术时使用最频繁的标签，它定义指定的目标节点被匹配时所需要应用的模板规则，并通过<xsl:apply-templates>标签激活并最终应用该模板规则定义的内容。

对象是当前标签节点中所有后继标签节点中的＜姓名＞标签元素。

2. ＜xsl:for-each＞标签元素语法及应用

＜xsl:value-of＞标签元素只能提取出一个节点的数据，而＜xsl:for-each＞标签元素允许循环处理被选择的节点集中的每个 XML 标签元素节点，它相当于编程语言中的循环控制语句。如果在该标签中应用了 select 属性，它将循环遍历由 select 属性值所指定的所有标签元素。如果添加选择数据的条件过滤运算符，还可以过滤输出的结果数据。

例 7-16 为利用＜xsl:for-each＞标签元素循环遍历例 7-10 描述软件学院学生信息的 XML 文档示例中的＜学生信息＞标签元素，并获得每个＜学生信息＞标签元素内的各个子标签的数据及标签属性值的代码示例。例 7-16 中的 XSLT 示例代码是对例 7-12 示例的简化，应用例 7-16 中的 XSLT 示例代码对例 7-10 示例中的 studentInfo. xml 文档进行样式控制后，在浏览器中预览的结果如图 7. 11 所示。

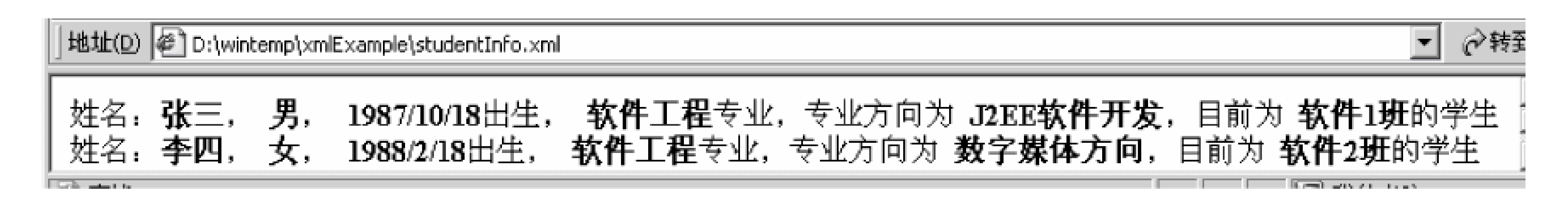


图 7. 11　对例 7-10 中的 XML 文档应用例 7-16 样式示例后在浏览器中预览的结果

例 7-16　体现＜xsl:for-each＞标签元素功能特性的 XSLT 代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                                                    version = "1.0">
    <xsl:template match = "/">
       <html>
            <head><title>北京蓝梦大学软件学院学生信息</title></head>
            <body><xsl:for - each select = "软件学院学生信息/学生信息">
                       姓名：<b><xsl:value - of select = "姓名"/></b>,
                       <b><xsl:value - of select = "@性别"/></b>,
                       <b><xsl:value - of select = "出生日期"/></b>出生,
                       <b><xsl:value - of select = "专业"/></b>专业,专业方向为
                       <b><xsl:value - of select = "./班级/@方向"/></b>,目前为
                       <b><xsl:value - of select = "班级"/></b>的学生<br/>
                 </xsl:for - each>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

＜xsl:for-each＞标签元素常用于匹配多个同名的标签节点，例 7-16 示例中的“软件学院学生信息/学生信息”是直接定位例 7-10 示例中的＜学生信息＞节点，并对它进行循环遍历，其中的“./班级/@方向”是定位在循环遍历过程中每个＜学生信息＞节点内的＜班级＞节点中的“方向”属性。

例 7-17 利用标签元素的属性值作为条件的 XSLT 代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                                                   version = "1.0">
<xsl:template match = "/">
  <html>
    <head><title>北京蓝梦大学软件学院学生信息</title></head>
    <body><xsl:for-each select = "软件学院学生信息/学生信息">
      <xsl:if test = "./班级[@方向 = 'J2EE 软件开发']">
        姓名:<i><b><xsl:value-of select = "姓名"/></b></i>,
        <i><b><xsl:value-of select = "@性别"/></b></i>,———— 以粗斜体格式显示输出
        <i><b><xsl:value-of select = "出生日期"/></b></i>出生,
        <i><b><xsl:value-of select = "专业"/></b></i>专业,专业方向为
        <i><b><xsl:value-of select = "./班级/@方向"/></b></i>,目前为
        <i><b><xsl:value-of select = "班级"/></b></i>的学生<br/>
      </xsl:if>
      <xsl:if test = "./班级[@方向 = '数字媒体方向']">
        姓名:<b><xsl:value-of select = "姓名"/></b>,
        <b><xsl:value-of select = "@性别"/></b>,———— 以粗体格式显示输出
        <b><xsl:value-of select = "出生日期"/></b>出生,
          <b><xsl:value-of select = "专业"/></b>专业,专业方向为
        <b><xsl:value-of select = "./班级/@方向"/></b>,目前为
        <b><xsl:value-of select = "班级"/></b>的学生<br/>
      </xsl:if>
    </xsl:for-each>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

例 7-18 中的 XSLT 代码示例只显示输出<学生信息>标签内的<班级>标签值为“软件 1 班”的学生信息,应用例 7-18 中的 XSLT 代码对例 7-10 示例中的 studentInfo.xml 文档进行样式控制后,在浏览器中预览的结果如图 7.13 所示。

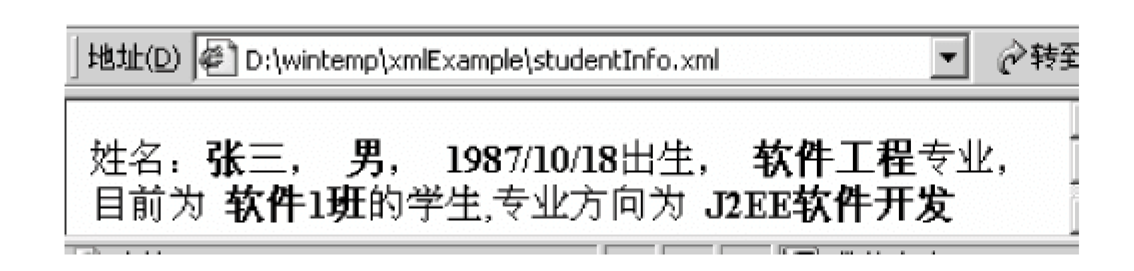


图 7.13 应用例 7-18 样式示例后在浏览器中预览的结果

例 7-18 利用标签元素的值作为条件的 XSLT 代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/TR/WD-xsl">
  <xsl:template match = "/">
    <html> <head><title>北京蓝梦大学软件学院学生信息</title></head>
```

templates>标签元素中，属于这些标签的子标签元素；另外，<xsl:sort>标签元素也不能应用在“http://www.w3.org/TR/WD-xsl”命名空间中，否则会出现如图 7.14 所示的错误，而只能应用在“http://www.w3.org/1999/XSL/Transform”命名空间中。

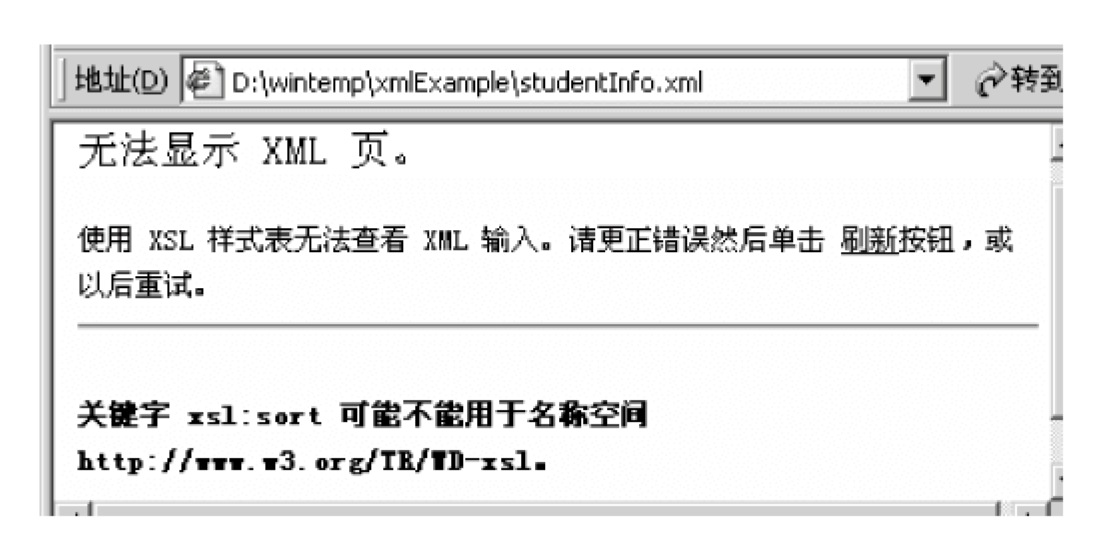


图 7.14　不正确地应用<xsl:sort>标签元素时产生的错误提示信息

（2）<xsl:sort>标签元素中的各个属性项目

<xsl:sort>标签元素包含了多个不同的属性项目，每个属性项目的可能取值及功能说明如下：

- select：为可选属性，指定需要进行排序操作的节点或节点组，默认为当前节点。
- data-type：为可选属性，指定排序的数据类型（text 和 number），默认为“text”文本数据类型。
- order：为可选属性，指定排序顺序（ascending 和 descending），默认为升序（ascending），中文字符按汉语拼音在字母表中的顺序排序。
- case-order：为可选属性，指定是否需要先按照字母的大小写顺序进行排序，取值为 upper-first 和 lower-first，默认为大写字符优先。

2. 利用<xsl:sort>标签元素排序 XML 文档中的标签数据值示例

例 7-20 是对例 7-16 示例中的<xsl:for-each>标签添加<xsl:sort>标签元素排序 XML 文档中的 0 标签数据值的代码示例，排序的节点为<学生信息>节点中的<姓名>节点，并且采用升序排序输出结果值（黑体所标识的标签）。

应用例 7-20 中的 XSLT 示例代码对例 7-10 示例中的 studentInfo.xml 文档进行样式控制后，在浏览器中预览的结果如图 7.15 所示。

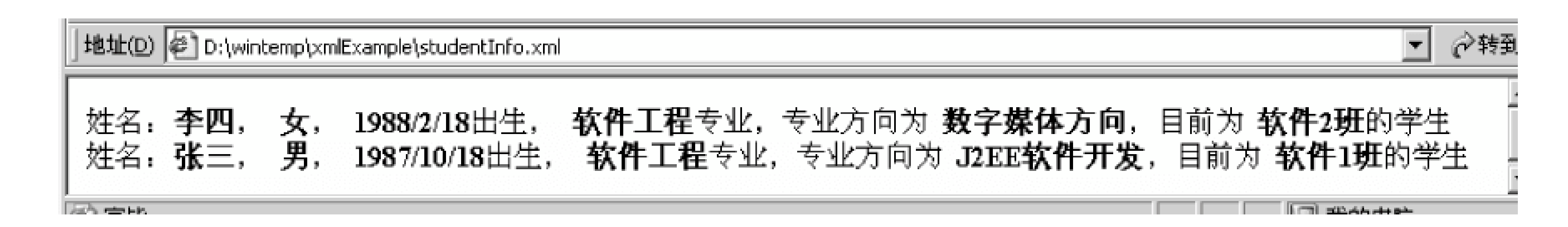


图 7.15　对例 7-10 的示例 XML 文档应用例 7-20 样式示例后在浏览器中预览的结果

例 7-20　对 XML 文档中的标签数据的输出结果进行排序的代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
```

注意<xsl:sort>标签元素的命名空间

```
                                        version = "1.0">
    <xsl:template match = "/">
      <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
           <body><xsl:for - each select = "软件学院学生信息/学生信息" >
                 <xsl:sort  select = "姓名"  order = "ascending" />  —— 对循环遍历的输出结果采用升序排序
               姓名:<b><xsl:value - of select = "姓名"/></b>,
               <b><xsl:value - of select = "@性别"/></b>,
               <b><xsl:value - of select = "出生日期"/></b>出生,
                  <b><xsl:value - of select = "专业"/></b>专业,专业方向为
               <b><xsl:value - of select = "./班级/@方向"/></b>,目前为
               <b><xsl:value - of select = "班级"/></b>的学生<br/>
           </xsl:for - each>
         </body>
      </html>
    </xsl:template>
  </xsl:stylesheet>
```

例 7-20 中的示例实现单一节点数据的排序,XSLT 也支持多节点数据,并且可以分别指定排序的节点顺序。例 7-21 所示的示例是利用<xsl:sort>标签元素(黑体所标识的标签)对例 7-10 示例中的 studentInfo. xml 文档实现多节点数据排序。

例 7-21 利用<xsl:sort>标签元素实现多节点数据排序示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                                        version = "1.0">
  <xsl:template match = "/">
    <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
        <body><xsl:for - each select = "软件学院学生信息/学生信息" >
               <xsl:sort select = "姓名"  order = "ascending" />
               <xsl:sort select = "班级" />
               <xsl:sort select = "专业/出生日期" />
               姓名:<b><xsl:value - of select = "姓名"/></b>,
               <b><xsl:value - of select = "@性别"/></b>,
                <b><xsl:value - of select = "出生日期"/></b>出生,
                <b><xsl:value - of select = "专业"/></b>专业,专业方向为
               <b><xsl:value - of select = "./班级/@方向"/></b>,目前为
               <b><xsl:value - of select = "班级"/></b>的学生<br/>
          </xsl:for - each>
        </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

在例 7-21 示例中,首先按照“姓名”升序排序,然后按“班级”排序,如果前两项排序的结果中出现相同的标签数据,再按“专业”和“出生日期”排序。

3. 利用<xsl:for-each>标签元素中的 order-by 属性实现排序功能

如果需要将<xsl:sort>标签元素应用在“http://www. w3. org/TR/WD-xsl”命名空

上，namespace 的值同样也可以为属性值模板动态创建命名空间字符串。

use-attribute-sets 表示一个通过空白分隔的属性集名称列表，指定此属性将在每个列出的属性集中声明每个属性。

2. <xsl:attribute>创建一个属性节点并将其附加到输出元素上

所有的<xsl:attribute>指令元素都必须要放在其父指令元素的任何其他内容之前，因为当已经开始输出元素内容之后，就不能再将附加的特性施加到标签元素中了。在对 XML 文档进行转换的过程中，通过应用<xsl:attribute>指令元素可以添加或修改某个标签元素中指定名称的属性。具体的应用参见例 7-24 所示的示例程序。下面为<xsl:attribute>指令元素的语法规则示例。

```
<xsl:attribute  name = "attribute - name"  namespace = "uri - reference" />
```

其中的 name 属性为必选项，表示要创建的属性项目的名称。如果此值为限定名称(QName)，属性节点同样也将在当前绑定到前缀上的命名空间中创建，除非 namespace 属性重写此值。name 属性的值也可以为一个属性值模板，动态获得属性名。

namespace 属性代表已创建属性的命名空间的统一资源标识符(URI)，如果在 name 属性中包含了前缀限定名(QName)，指定的前缀将绑定到由 namespace 属性所指定的命名空间上，namespace 的值同样也可以为属性值模板而动态创建命名空间字符串。

3. <xsl:attribute-set>定义命名的属性集

<xsl:attribute-set>指令元素实现在样式表的顶层标签元素中定义属性集和指定属性集中的各个属性成员，从而可以实现将属性分组。然后就可以在<xsl:element>等指令元素中通过 use-attribute-sets 属性应用该属性集，实现对该属性分组的重用，减少对属性项目的重复定义。下面为<xsl:attribute-set>指令元素的语法规则示例。

```
<xsl:attribute - set  name = attributeName use - attribute - sets = attributeNames />
```

其中的 name 属性为必选项，表示该属性集的限定名。而 use-attribute-sets 属性为一个通过空白分隔的属性集名称的列表。

在<xsl:attribute-set>指令元素中必须要包含用于组成指定属性集中的成员属性的<xsl:attribute>指令元素。当然，要应用指定名称的属性集，可以在<xsl:element>、<xsl:copy>或<xsl:attribute-set>等指令元素中通过它们的 use-attribute-sets 属性指定某个名称的属性集。

如下代码片段示例定义了一个名称为“titleStyle”的属性集，在该属性集中包含了两个属性成员，分别为 fontSize(代表字体大小)和 fontWeight(代表粗体字形)。

```
<xsl:attribute - set name = "titleStyle" >
     <xsl:attribute name = "fontSize"> 12pt </xsl:attribute>
    <xsl:attribute name = "fontWeight"> bold </xsl:attribute>
</xsl:attribute - set>
```

4. 动态生成 XML 标签元素的 XSLT 代码示例

在基于 XML 的应用系统开发中，为了能够更好地浏览 XML 文档中的数据，经常需要将 XML 文档从一种格式转换为另一种格式，包括将 XML 转换为 XHTML 格式。

例 7-23 中的代码为描述某个课程信息的 XML 文档示例，该 XML 文档保存在文件名为“courseInfo. xml”的文件中。在其中黑体所标识的样式表声明指令中，指定本 XML 文档所对应的 XSLT 文件为“courseInfo. xsl”。

例 7-23 描述某个课程信息的 XML 文档示例

```
<?xml version = "1.0" encoding = "gb2312"?>
<?xml - stylesheet type = "text/xsl" href = "courseInfo.xsl"?>
<courseInfos>
   <oneCourseInfos>
    <teacherName>张三</teacherName>
    <teacherPhotoURL>zhang.gif</teacherPhotoURL>
    <url>http://www.bluedream.com/</url>
    <courseName>J2EE 系统架构及设计模式</courseName>
  </oneCourseInfos>
  <oneCourseInfos>
  <teacherName>李四</teacherName>
    <teacherPhotoURL>li.gif</teacherPhotoURL>
    <url>http://www.bluedream.com/</url>
    <courseName>XML 应用开发提高</courseName>
  </oneCourseInfos>
</courseInfos>
```

例 7-24 中的代码为例 7-23 中的 XML 文档所对应的 XSLT 文件“courseInfo. xsl”中的代码示例。在该 XSLT 文件中动态地创建出另一种满足浏览器浏览格式要求的 XML 文档，从而最终达到对例 7-23 中的 XML 文档示例进行格式转换的目的。

例 7-23 示例中的 courseInfo. xml 文件在浏览器中预览的最终结果如图 7. 16 所示，由于转换为 XHTML 格式的<a>超链接和<img>图片标签，在浏览器中将能够正常预览。



图 7. 16 例 7-23 示例中的 XML 文档在浏览器中预览的最终结果

例 7-24 XSLT 文件 courseInfo. xsl 中的代码示例

```
<?xml version = "1.0" encoding = "gb2312"?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                                                     version = "1.0" >
  <xsl:output method = "xml" indent = "yes" encoding = "gb2312"/>
```

```
  <xsl:template match = "/">
    <xsl:apply - templates />
  </xsl:template>
  <xsl:template match = "oneCourseInfos">
    <b><xsl:value - of select = "courseName" /></b>老师的详细信息:
    <xsl:element name = "a">
      <xsl:attribute name = "href"><xsl:value - of select = "url" />
      <xsl:value - of select = "teacherName" /></xsl:attribute>
      <xsl:attribute name = "target">_blank</xsl:attribute>
      <xsl:value - of select = "teacherName" />
    </xsl:element>
    <xsl:element name = "img">
        <xsl:attribute name = "src">
        <xsl:value - of select = "teacherPhotoURL" /></xsl:attribute>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

动态创建 XHTML 中的<a>标签元素和设置其中的各个成员属性

动态创建 XHTML 中的<img>标签元素和设置其中的各个成员属性

在例 7-24 的 XSLT 文件示例中,动态地创建出满足浏览器浏览要求的<a>和<img>标签元素,但这些标签元素的属性和正文值都来源于原始的 XML 文档。

5. 动态生成 HTML 标签元素的 XSLT 代码示例

XML 文档可以保存应用系统中有价值的系统配置数据,应用系统根据保存在 XML 文档中的配置数据动态地创建出应用系统的用户界面,从而使得应用系统的用户界面是可以配置的"动态界面"。

例 7-25 中的代码为一个描述 Web 表单调整信息的 XML 文档代码示例。为了简化问题,在该 XML 文档中只给出一个按钮的描述信息(按钮的类型和按钮的 id 名),并保存在文件名为"createHTMLByXSLT.xml"的 XML 文件中。在其中黑体所标识的样式表声明指令中,指定本 XML 文档所对应的 XSLT 文件为"createHTMLByXSLT.xsl"。

例 7-25　描述 Web 表单的 XML 文档代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<?xml - stylesheet href = "createHTMLByXSLT.xsl" type = "text/xsl" ?>
< input inputType = "reset" id = "inputElementID" />
```

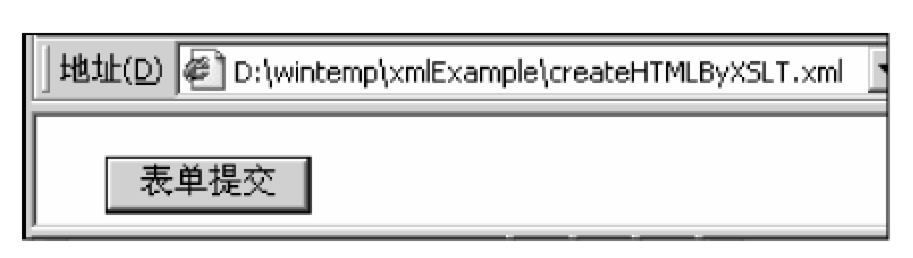
例 7-26 中的代码为例 7-25 中的 XML 文档所对应的 XSLT 扩展样式表文件"createHTMLByXSLT.xsl"中的代码示例。在该 XSLT 文件中依据原始的 XML 文档中的特征描述信息动态地创建出 HTML 表单及表单中的控件元素,但数据都来源于原始的 XML 文档。

例 7-26　XSLT 文件 createHTMLByXSLT.xsl 中的代码示例

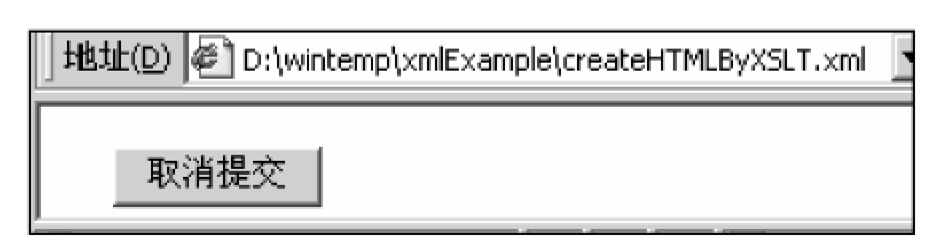
```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                version = "1.0" >
```

```
<xsl:output method="html"          指定输出的文档类型为 HTML
        indent="yes" encoding="gb2312"/>
<xsl:template match="/">
  <html><head><title>利用 XSLT 将 XML 转换为 HTML 示例</title></head>
     <body><xsl:apply-templates select="/input" /></body>
   </html>                                          指定需要转换的 XML
</xsl:template>                                     文档中的标签节点
<xsl:template match="/input">
  <xsl:element name="input">      创建 HTML 中的<input>按钮标签
     <xsl:attribute name="type">
        <xsl:value-of select="@inputType" />
     </xsl:attribute>
     <xsl:choose>                                       依据配置信息中的按钮类型
        <xsl:when test="@inputType='reset'">            值动态设定按钮的类型
          <xsl:attribute name="value">取消提交</xsl:attribute>
       </xsl:when>
       <xsl:when test="@inputType='submit'">
         <xsl:attribute name="value">表单提交</xsl:attribute>
       </xsl:when>
    </xsl:choose>            依据配置信息中的 id 值动态设定按钮的 id 属性值
    <xsl:attribute name="id"><xsl:value-of select="@id" /></xsl:attribute>
    <xsl:if test="@name">
<xsl:attribute name="name"><xsl:value-of select="@id" /></xsl:attribute>
     </xsl:if>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

例 7-25 示例中的 XML 文档在浏览器中预览的结果如图 7.17(a)和图 7.17(b)所示，其中图 7.17(a)为例 7-25 示例 XML 文档中的<input>标签内的 inputType 属性值为 submit 时的结果(inputType="submit"，表示提交类型的按钮)；而图 7.17(b)为例 7-25 示例 XML 文档中的<input>标签内的 inputType 属性值为 reset 时的结果(inputType="reset"，表示重置类型的按钮)。



(a) 动态创建出提交类型的按钮



(b) 动态创建出重置类型的按钮

图 7.17

6. 基于 XML 数据动态创建用户界面的 XSLT 代码示例

目前流行的动态网站开发技术主要有 JSP、ASP、ASP.NET、PHP 等形式，但这些动态网站开发技术都与后台服务器端的实现技术有关。其实利用“XML+XSLT+XHTML”也可以实现动态网站开发技术，而且与服务器端的技术实现平台无关。如果 XML 文档由服务器端程序预先动态创建好，并传送到客户端，还可以直接在客户端动态创建，达到离线操

作的效果。

例 7-27 中的代码为一个描述蓝梦在线即时通信系统用户信息的 XML 文档代码示例。为了简化问题，只附录了少量的数据，并保存在文件名为“goodFriends. xml”的 XML 文件中。其实该 XML 文件中的数据可以由服务器端程序动态创建。

在其中黑体所标识的样式表声明指令中，指定本 XML 文档所对应的 XSLT 文件为“goodFriends. xsl”。

例 7-27 描述蓝梦在线即时通信系统用户信息的 XML 文档代码示例

```
<?xml version="1.0" encoding="gb2312"?>
<?xml-stylesheet type="text/xsl" href="goodFriends.xsl"?>
<goodFriends>
  <oneFriend>
    <friendID>1</friendID><friendName>网上张飞</friendName>
  </oneFriend>
  <oneFriend>
    <friendID>2</friendID><friendName>黑脸关公</friendName>
  </oneFriend>
  <oneFriend>
    <friendID>3</friendID><friendName>诸葛孔明</friendName>
  </oneFriend>
</goodFriends>
```

例 7-28 中的代码为例 7-27 中的 XML 文档所对应的 XSLT 扩展样式表文件“goodFriends. xsl”中的代码示例。在该 XSLT 文件中依据原始的 XML 文档中的特征描述信息动态地创建出系统的操作界面(只简单地创建出超链接)，但数据都来源于原始的 XML 文档。例 7-27 示例中的 XML 文档在浏览器中预览的结果如图 7. 18 所示。

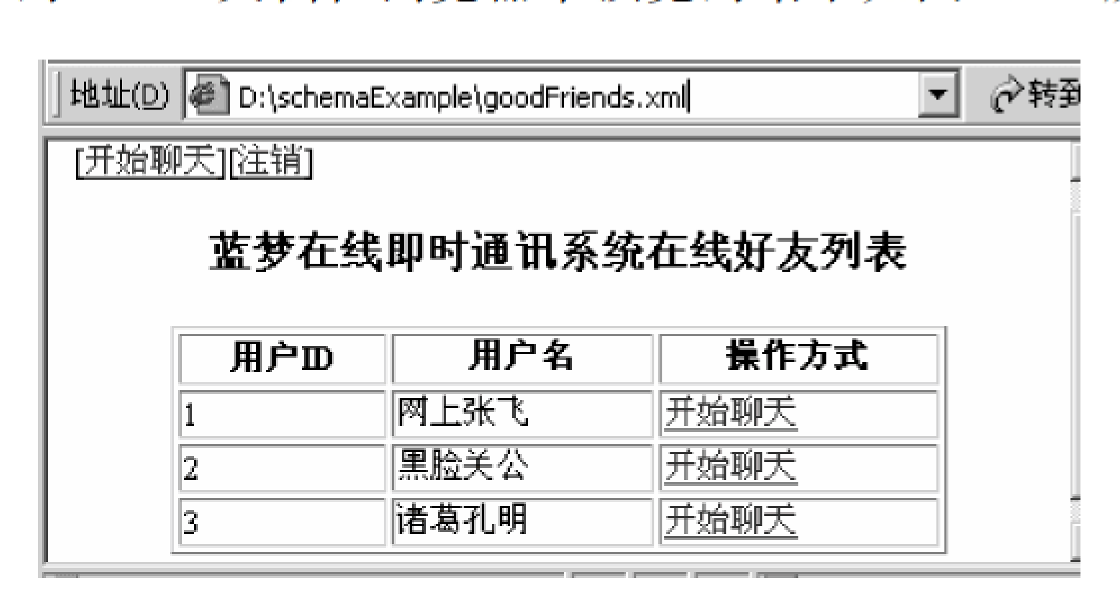


图 7. 18 例 7-27 示例中的 XML 文档预览的结果

例 7-28 XSLT 文件 goodFriends. xsl 中的代码示例

```
<?xml version="1.0" encoding="gb2312"?>
<xsl:stylesheet version="1.0"
                    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output method="html" indent="yes"/>
<xsl:template match="/">
    <html><head><title>蓝梦在线即时通信系统在线好友列表</title></head><body>
```

```
    function changeXSLT(xsltFileName){
        var xmlObject = new ActiveXObject("Microsoft.XMLDOM")
        xmlObject.async = false
        xmlObject.load("goodFriends.xml")                    // 加载 XML 文档
        var xslObject = new ActiveXObject("Microsoft.XMLDOM")
        xslObject.async = false
        xslObject.load(xsltFileName)                         //加载 XSL 文档
        document.write(xmlObject.transformNode(xslObject))   //完成转换
    }
  </script>
</head><body><form action = "#" method = "post" >请选择所希望的 XSLT:
     <select  onchange = "changeXSLT(options[selectedIndex].value);">
    <option selected = "selected" value = "goodFriends.xsl">表格形式</option>
    <option value = "stringFriends.xsl">文字形式</option>
    <option value = "imageFriends.xsl">图形方式</option>
  </select></form>  </body></html>
```

当操作者在如图 7.19 所示的下拉列表框中选择某种风格的 XSLT 项目后，将触发一个 onchange 事件，同时也将执行 changeXSLT()事件响应函数。在该函数中动态获得目标 XSLT 文件(本示例采用例 7-28 中的“goodFriends.xsl”文件)，见黑体所标识的函数代码。

在 changeXSLT()事件响应函数代码中，首先创建微软 IE 浏览器中内置的 XML 解析器组件的一个对象实例(对象名称为“xmlObject”)，然后把目标 XML 文件(本示例采用例 7-27 示例中的“goodFriends.xml”文件)载入到内存中；然后创建 IE 浏览器中内置的解析器组件的另一个对象实例，实现把目标 XSL 文件载入到内存中；最后利用解析器组件对象中的 transformNode()方法使用所选择的 XSL 文档转换 XML 文档中的数据，并在当前浏览器中把转换后的结果作为 XHTML 页面显示输出。

本章小结

教学重点

作者在本章的最前面开门见山地提出了若干个问题，其实探究这些问题的答案所涉及的相关知识，以及熟练地掌握这些知识相关的技术实现，其构成本章的教学重点。

因此，XSLT 技术特性及应用、XSLT 中的模板规则定义和激活技术，以及 XSLT 输出及流程控制元素等方面的内容就是本章的主要教学内容。

学习难点

XSLT 其实是一种函数式的语言(Functional Language)，而不是过程性语言，在 C 或 Pascal 等过程性语言中，程序其实是被定义成一系列的功能实现步骤而形成程序指令流。这些步骤按照规定的顺序执行，并在最后一步产生最终的处理结果。而在 XSLT 等这样的

（2）下面哪一项是对<xsl:template>标签主要功能的正确描述？

A. 定义一个模板语言，输出 HTML 标签

B. 定义一个模板规则，而转换规则的匹配模式则由属性 match 来指定

C. 定义一个临时缓冲区，缓存 XML 文档数据

D. 定义一个模板规则，只能输出 HTML 标签

（3）下面哪一项中的 XSLT 指令元素可以表示单条件判断？

A. <xsl:if>指令元素　　B. <xsl:choose>指令元素

C. <xsl:value-of>指令元素　　D. <xsl:for-each>指令元素

（4）下面哪一项中的 XSLT 指令元素可以产生排序效果？

A. <xsl:if>指令元素　　B. <xsl:element>指令元素

C. <xsl:sort>指令元素　　D. <xsl:for-each>指令元素

（5）下面哪一项中的 XSLT 指令元素可以动态生成 XML 标签元素？

A. <xsl:variable>指令元素　　B. <xsl:element>指令元素

C. <xsl:sort>指令元素　　D. <xsl:output>指令元素

2. 填空题

（1）XSL 代表________，主要包含________和________，它们都是________颁布的技术标准。

（2）W3C 为什么要发布 XSL 技术规范，是因为 CSS 存在如下几个主要的缺点：________、________和________。

（3）请仔细阅读例 7-5 中的 XSLT 文档示例，完成下面的填空。

该 XSLT 文档示例中的<xsl:stylesheet>表示________，其中的 version="1.0"表示________，而<xsl:template match="/">中的 match="/"表示________，<xsl:value-of select="firstXSLT"/>中的 select="firstXSLT"表示________。

（4）XSLT 中的<xsl:include>指令元素的主要作用是________，<xsl:import>指令元素的主要作用是________，<xsl:output>指令元素的主要作用是________，<xsl:variable>指令元素的主要作用是________。

（5）XSLT 中的<xsl:element>指令元素的主要作用是________，<xsl:attribute>指令元素的主要作用是________，<xsl:attribute-set>指令元素的主要作用是________。

3. 问答题

（1）为什么要提供 XSL 技术？XSL 包含哪两部分？请描述 XSLT 的主要作用。

（2）用一个简单的 XML 文档示例来描述 XSLT 的工作过程。请描述 XSLT 中的模板是什么，模板一般由哪两部分的语句所组成？

（3）请描述在 XSLT 中“选择模式”的各种语句所能够完成的功能，有哪些具体的语句。

（4）请描述在 XSLT 中“测试（识别）模式”的各种语句所能够完成的功能，有哪些具体的语句。

（5）请描述在 XSLT 中“匹配模式”的各种语句所能够完成的功能，有哪些具体的语句。

第 8 章　XML XPath 技术及应用

CHAPTER 8

XPath(eXtensible Path,可扩展路径)是一种专为查询 XML 文档而设计的查询语言,它基于 XML 文档的逻辑结构,并在该结构中进行导航。而且 XPath 和 XSLT 总是相互配合使用,能够快速、准确地定位目标节点,但 W3C 将它们分成两个不同的标准。因此,XPath 本身是一个独立的规范,它不仅可以与 XSLT 相互配合使用,也可以与 XPointer、XLink 和 XQuery 相互配合使用。

XPath 除了提供一套定位语法之外,还提供了功能丰富的标准内置函数、多种数据类型及数值运算、布尔运算和字符串处理等方面的功能。尽管 XPath 有很强的表达能力,但 XPath 本身并不是一种完整的编程语言。

如果将 XPath 和某种编程语言(如 Java)相互结合应用,则能够发挥各自的技术特性。比如,用 XPath 定义查询目标,用 Java 语言编写复杂的查询逻辑。为此,Java 5 提供了 javax.xml.xpath 软件包,为 Java 平台的开发人员提供一个引擎和对象模型独立的 XPath 系统库。

本章将系统地介绍 XML XPath 技术及应用,包括 XPath 技术特性及语法、XPath 中各种形式的操作符及应用、XPath 中的功能函数及应用、Java 平台对 XPath 技术的支持。

8.1　XPath 技术特性及语法

8.1.1　XPath 技术特性及应用

1. XPath 技术概述

(1) XPath 是什么

XPath 是一种专门用来在 XML 文档中查找节点信息的语言,节点是通过沿着路径(Path) 或者步(Step)被选取的。XPath 和 XSLT 一起于 1999 年 11 月 16 日成为 W3C 推荐的标准。在 W3C 的官方网站(http://www.w3.org/TR/xpath/)上提供了对 XPath 技术规范的详细介绍信息,如图 8.1(a)所示。在国内有一个名为“开放论坛”的网站(http://www.opendl.com/openxml/w3/TR/xpath/xpath-

为 XSLT 在转换 XML 文档过程中需要使用 XPath 定位 XML 层次树中的节点。

XPath 基于 XML 文档的逻辑结构，并在该结构中进行导航。比如，若不应用 XPath 技术，希望访问一个 XML 文档中名称为“teacher”标签中的 workUnit（工作单位）属性值时，就必须要使用以下形式的 XSLT 指令。

```
<xsl:for-each select="teacher">
    <xsl:value-of select="@workUnit"/>
</xsl:for-each>
```

但如果在 XSLT 指令中应用 XPath 之后，就可以采用如下更简单的 XSLT 指令。

```
<xsl:value-of select="teacher/@workUnit"/>
```

4. XPath 中的主要节点类型及说明示例

（1）XML 文档逻辑结构的基本组成要素

一个 XML 文档可以包含标签元素、CDATA 段、注释、处理指令等逻辑结构的组成要素，其中的标签元素还可以包含不同的属性。在 XPath 的技术规范中，将 XML 文档中各种形式的节点划分为 7 种不同的类型：标签元素、属性、文本、命名空间、处理器指令、注释及文档节点（或根节点）。

（2）XPath 中的主要节点类型及说明示例

例 8-1 是描述 J2EE 架构与程序设计必修课信息的 XML 文档，并通过标注指示出不同类型的节点。

例 8-1　描述 J2EE 架构与程序设计必修课信息的 XML 文档示例

```
<?xml version="1.0" encoding="GB2312"?>——处理器指令节点
<?xml-stylesheet type="text/xsl" href="courseInfo.xsl"?>——XPath 中的根节点
<courseInfos>
  <oneCourseInfo>——标签元素节点
    <courseName>J2EE 架构与程序设计</courseName>——属性节点
    <teacher workUnit="中科院计算所">杨老师</teacher>——文本节点
    <courseTimes>48 学时</courseTimes>
    <classRoom>9 教东 102</classRoom>
    <weekTimes>3</weekTimes>
    <instruction>本课程的主要内容包括……</instruction>
    <testMethod homeWork="6">
        <lastHomeWork>利用所学习的 J2EE Web 技术设计和实现 XXXX 系统</lastHomeWork>
        <test>在课程结束进行考试</test>
     </testMethod>
     <!-- 该课程的学分为 3 学分 -->——注释节点
  </oneCourseInfo>
</courseInfos>——根标签，也被称为文档节点
```

6. XPath 中的各种数据类型

XPath 中的表达式被处理之后将产生一个对象，这种对象的数据类型为以下 4 种数据类型之一。

(1) 节点集(Node-Set)

在应用 XPath 时，节点集是通过路径匹配返回的符合条件的一组节点的集合，其他类型的数据不能转换为节点集。

(2) 布尔值(Boolean)

由 XPath 函数或布尔表达式返回的匹配条件的值，与编程语言中的布尔值类同，并且也有 true 和 false 两个不同的值。在 XPath 中，布尔类型的值可以与数值类型、字符串类型的值进行相互转换。

(3) 字符串(String)

在 XPath 中提供了一系列的与字符串功能处理相关的函数，与编程语言中的字符串类同，字符串也可与数值类型、布尔值类型的数据相互转换。

(4) 数值(Number)

在 XPath 中数值为双精度 64 位浮点数，而且在 XPath 中提供了代表一些特殊数值的符号常量，如非数值 NaN(Not-a-Number)、正无穷大(infinity)、负无穷大(-infinity)等。XPath 中的数值也可以与布尔类型、字符串类型相互转换。

8.1.2 XPath 中的节点定位语法

1. XPath 中的节点定位方式

在对 XML 标签节点数据转换中如何定位目标节点或节点集？在 XPath 中可以采用路径匹配、位置匹配、属性及属性值匹配、亲属关系匹配(利用节点之间的层次关系和归属关系)和条件匹配等形式进行节点定位。

这些节点定位的表达式主要出现在模板声明指令<xsl:template>中的 match 属性和模板应用指令<xsl:apply-templates>中的 select 属性中。

2. XPath 中的路径匹配节点应用示例

XPath 中路径定位方式的语法类似于操作系统中文件系统的文件定位，并且位置路径字符串中均包括一个或多个步，每个步都被斜杠分割。只要熟悉操作系统中的文件定位方法，就能够快速掌握 XPath 的路径定位语法。下面以例 7-10 所示的描述某个软件学院的学生信息的 XML 文档为示例，说明 XPath 中路径匹配节点的语法及定位方式。

(1) 绝对路径定位方式下的根节点表示形式

在应用路径匹配定位节点时，由于路径定位方式可以是相对路径(Relative Location Path)形式，也可以是绝对路径(Absolute Location Path)形式。绝对定位路径是以斜线"/"开头，而相对定位路径由"/ "分开的一个或多个路径组成。如果 XPath 中的路径定位方式

```
</xsl:stylesheet>
```

3. XPath 中的位置匹配节点应用示例

位置匹配主要是利用 XPath 中获得标签节点位置的功能函数 last()和 position()匹配目标节点。下面仍然以例 7-10 所示的描述某个软件学院的学生信息的 XML 文档为示例，说明 XPath 中的位置匹配节点的语法及定位方式。

(1) 匹配一组标签中的第一个标签节点

在匹配的条件表达式中如果直接指示位置号为 1，则表示匹配一组标签中的第一个标签节点。如示例：<xsl:apply-templates select ="学生信息[1]"/>，用来匹配例 7-10 示例文档中的第一个<学生信息>标签元素节点。

(2) 匹配一组标签中的最后一个标签节点

在匹配的条件表达式中，如果应用 last()函数，则表示匹配一组标签中的最后一个标签节点。如示例：<xsl:apply-templates select ="学生信息[last()]"/>，用来匹配例 7-10 示例文档中的最后一个<学生信息>标签元素节点。

(3) 匹配一组标签中指定位置的标签节点

在匹配的条件表达式中，如果应用 position()函数，并规定具体的位置值，则表示匹配一组标签中指定位置的标签节点。如示例：<xsl:apply-templates select ="学生信息[position()=2]"/>，用来匹配例 7-10 示例文档中的第二个<学生信息>标签元素节点。

例 8-3 为应用 XPath 中的位置匹配方式定位例 7-10 示例 XML 文档中的节点的应用示例，并注意其中黑体所标识的代码。

例 8-3 位置匹配节点的应用示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                                                       version = "1.0">
    <xsl:template match = "/">
        <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
            <body><xsl:apply - templates select  = "软件学院学生信息"/></body>
        </html>
    </xsl:template>
    <xsl:template match = "软件学院学生信息">
        <xsl:apply - templates select  = "学生信息[position() = 2]"/>    匹配第二个<学生信息>标签元素节点
    </xsl:template>
</xsl:stylesheet>
```

4. XPath 中的属性及属性值匹配节点应用示例

属性匹配主要是利用 XPath 中的属性访问符“@”限定属性名(属性名前要有“@”前缀符)，并设置具体的条件表达式匹配目标节点。下面仍然以例 7-10 所示的描述某个软件学院的学生信息的 XML 文档为示例，说明 XPath 中属性匹配节点的语法及定位方式。

（2）XPath 中的关系运算符

关系运算符主要有＝（相等）、!＝（不等）、＜（小于）、＜＝（小于等于）、＞（大于）、＞＝（大于等于），利用这些关系运算符很容易构建出各种条件表达式。

（3）XPath 中的数值运算符

数值运算符类似于编程语言中的算术运算符，主要包括＋（加）、－（减）、*（乘）、div（除）和 mod（取模）。

对于其他的运算符，可以参考 XPath 的技术文档进一步了解。正是由于 XPath 提供了丰富的运算符，才使得 XPath 也能够表达比较复杂的逻辑关系。如示例：<xsl:value-of select="//学生信息[position() mod 2 = 0]"/>，表示选择偶数位置的＜学生信息＞标签元素。

2．XPath 中具有节点匹配功能的主要操作符及对应的功能描述

在 XPath 中也提供了多种形式的操作符，灵活地应用这些操作符能够设计出复杂的匹配条件表达式。比如，通过在匹配条件表达式中灵活地应用“|”操作符，可以同时匹配多个不同的路径表达式的节点。

下文列出的各个示例是以例 7-10 所示的描述软件学院学生信息的 XML 文档为原型示例文档，通过示例说明典型操作符的应用。

（1）“/”操作符的功能描述

“/”表示父子关系，并匹配子标签元素节点。如果“/”位于最左侧，则表示选择根节点（和 UNIX 系统中关于文件路径的定义是一致的）内的直接子标签元素。如示例：“/软件学院学生信息”表示从示例文档的根节点开始选取；而示例：studentInfos/studentInfo 表示以标签元素＜studentInfos＞为祖先的所有＜studentInfo＞标签元素节点。

（2）“//”操作符的功能描述

“//”表示从匹配选择的当前节点中选择文档中的指定节点，而不考虑它们的位置。如果位于最左侧，表示从根节点出发递归下降搜索到指定标签元素。如示例：“//学生信息”表示选取所有＜学生信息＞子标签元素节点，而不管它们在文档中的位置；而示例：“软件学院学生信息//学生信息”表示选择所有属于＜软件学院学生信息＞标签元素节点后代的＜学生信息＞标签元素节点，而不管它们位于＜软件学院学生信息＞标签元素节点之下的什么位置。

（3）“.”操作符的功能描述

“.”表示当前标签元素节点（正在处理的节点）

（4）“..”操作符的功能描述

“..”表示当前节点的父节点。

（5）“指定节点名”操作符的功能描述

选取此节点名的所有子节点，如示例：“软件学院学生信息”表示选取＜软件学院学生信息＞标签元素节点的所有子节点。

（6）“*”操作符的功能描述

“*”为通配符，表示匹配任意标签元素节点，可用来选取未知的 XML 元素。如示例：

- 示例：//出生日期 | //班级

选取例 7-10 示例 XML 文档中所有“出生日期”和“班级”标签元素节点值。

- 示例：//学生信息/出生日期| //专业

选取例 7-10 示例 XML 文档中所有属于“学生信息”标签元素的“出生日期”标签元素节点值，以及 XML 文档中所有的“专业”标签元素节点值。

例 8-6 为应用 XPath 中的逻辑或操作符构建匹配表达式定位例 7-10 示例 XML 文档中的节点的应用示例，并注意其中黑体所标识的代码。

例 8-6 XPath 中的逻辑或操作符的应用示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" version = "1.0">
  <xsl:template match = "/">
    <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
        <body><xsl:apply-templates select  = "软件学院学生信息"/></body>
    </html>
  </xsl:template>
  <xsl:template match = "软件学院学生信息">
    <xsl:apply-templates select  = "//学生信息/姓名 | //学生信息/专业"/><br/>
    <xsl:apply-templates select  = "//出生日期 | //班级"/><br/>
    <xsl:apply-templates select  = "//学生信息/出生日期 | //专业"/><br/>
  </xsl:template>
</xsl:stylesheet>
```

8.2.2 XPath 中的条件匹配表达式及谓语

1. XPath 中的条件表达式及应用

（1）应用条件表达式匹配目标节点

在应用 XSLT 技术时，不仅需要应用 XPath 中的各种路径指示符匹配 XML 文档中的目标节点，还需要应用条件表达式匹配符合条件的目标节点，并过滤或排序 XML 文档中的节点数据。为此需要应用 XPath 中的谓语(Predicates)。

（2）XPath 中的谓语

XPath 中的谓语是用来查找某个特定的节点或者包含某个特定值的节点，谓语被嵌在方括号([ ])中形成路径表达式或条件表达式，如示例“//学生信息[姓名="张三"]”所示。通过应用谓语可以更细致地提炼所选中的目标节点集，并获得期望的结果。

2. XPath 中的谓语及条件表达式的各种应用示例

在谓语中可以使用各种函数、定位路径表达式、数值表达式、关系表达式等形式进一步指定某个符合条件的元素。下面仍然以例 7-10 所示的描述软件学院学生信息的 XML 文档为示例，说明 XPath 中的谓语及条件表达式的各种应用。

（1）示例：/软件学院学生信息/学生信息[1]

选取属于＜软件学院学生信息＞子标签元素中的第一个＜学生信息＞标签元素，其中的数值1代表＜学生信息＞节点集中的第一个节点。

（2）示例：/软件学院学生信息/学生信息[last()]

选取属于＜软件学院学生信息＞子标签元素中的最后一个＜学生信息＞标签元素，其中的last()为XPath函数，返回正在处理的节点集中的最后一个节点的位置值。

（3）示例：/软件学院学生信息/学生信息[last()－1]

选取属于＜软件学院学生信息＞子标签元素中的倒数第二个＜学生信息＞标签元素，其中的“last()－1”为数值表达式。

（4）示例：/软件学院学生信息/学生信息[position()＜3]

选取最前面的两个属于＜软件学院学生信息＞标签元素中的＜学生信息＞子标签元素，其中的position()为XPath函数，返回当前节点正在处理的节点集中的位置，“position()＜3”为条件表达式。

（5）示例：//班级[@方向]

选取所有拥有名为“方向”属性的＜班级＞标签元素。

（6）示例：//班级[@方向＝'数字媒体方向']

选取所有＜班级＞标签元素，并且这些标签元素的“方向”属性值为“数字媒体方向”。

（7）示例：/软件学院学生信息/学生信息[姓名＝'张三']

选取所有＜软件学院学生信息＞标签元素中的＜学生信息＞标签元素，并且其中的＜姓名＞标签元素的值为“张三”。

（8）示例：/软件学院学生信息/学生信息[姓名＝'张三']/专业

选取所有＜软件学院学生信息＞标签元素中的＜学生信息＞标签元素内的＜专业＞标签元素，并且＜学生信息＞标签元素内的＜姓名＞标签元素的值为“张三”。

3．XPath中的条件匹配节点的应用示例

下面仍然以例7-10所示的描述软件学院学生信息的XML文档为示例，说明XPath中各种形式的条件匹配节点的应用，其中也应用带有谓语的条件表达式。

（1）识别子标签元素是否存在

示例：//学生信息[姓名]

识别在例7-10示例XML文档中的“学生信息”标签元素下是否存在“姓名”子标签元素，如果存在，则选择包含有＜姓名＞子标签元素的所有＜学生信息＞标签元素。

（2）获取子标签元素的值

示例：//学生信息[姓名＝'张三']

选择例7-10示例XML文档中含有＜姓名＞子标签元素，并且“姓名”标签的取值为“张三”的所有“学生信息”标签。

（3）识别标签中的特定属性项目是否存在

示例：//学生信息[@性别]

识别在例7-10示例XML文档中的“学生信息”标签元素中是否包含有“性别”属性项目，如果存在这样的“学生信息”标签元素，则选择这些包含有“性别”属性的＜学生信息＞标

```
        <td><xsl:value-of select="@homeWork"/></td>
        <td><xsl:value-of select="lastHomeWork"/></td>
        <td><xsl:value-of select="test"/></td>
</xsl:template>
</xsl:stylesheet>
```

(3) 将授课老师为“谢老师”所讲的各个课程信息以某种形式的表格显示输出

例 8-10 中的代码示例是满足该功能要求的 Courses. xsl 文件,在输出的 HTML 标签中应用表格<table>标签以行、列的形式显示课程信息中的有关数据。此时例 8-8 中的 XML 文档示例代码在浏览器中的执行结果如图 8.3 所示。

地址(D) D:\schemaExample\Courses.xml 转到

下面为2010年软件学院部分课程信息表

课程名称	老师	老师的工作地	课时	上课教室	每周课次	介绍	测试方式	作业次数	最后的作业	测试
软件系统分析与设计技术实践	谢老师	中科院计算所	32学时	思源301	1	本课程是...	设计和实现XXXX系统 在课程结束进行考试	8	设计和实现XXXX系统	在课程结束进行考试
软件需求工程	谢老师	中科院计算所	32学时	思源401	1	本课程是...	设计和实现XXXX系统 在课程结束进行考试	8	设计和实现XXXX系统	在课程结束进行考试

图 8.3　例 8-8 中的 XML 文档示例代码在浏览器中的执行结果

例 8-10　查询谢老师所讲的各个课程信息的 Courses. xsl 文件代码示例

```
<?xml version="1.0" encoding="gb2312" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
    <html><head><title>2010 年软件学院部分课程信息表</title></head>
        <body><xsl:apply-templates/></body>
    </html>
</xsl:template>
<xsl:template match="Courses">
    <center><h3>下面为 2010 年软件学院部分课程信息表</h3></center>
    <table border="1">
        <th>课程名称</th><th>老师</th><th>老师的工作地</th><th>课时</th>
        <th>上课教室</th><th>每周课次</th><th>介绍</th><th>测试方式</th>
        <th>作业次数</th><th>最后的作业</th><th>测试</th><xsl:apply-templates/>
    </table>
</xsl:template>
<xsl:template match="Course">
    <tr><!-- 注意在下面" = "的两旁留空格,要不然会显示错误 -->
        <xsl:if test=".[teacher = '谢老师']">
            <td><xsl:value-of select="courseName"/></td>
            <td><xsl:for-each select="teacher"><xsl:value-of />
                 </xsl:for-each></td>
             <xsl:apply-templates select="teacher"/>
            <td><xsl:value-of select="courseTimes"/></td>
            <td><xsl:value-of select="classRoom"/></td>
```

```
            <td><xsl:value-of select="weekTimes"/></td>
            <td><xsl:value-of select="instruction"/></td>
            <td><xsl:for-each select="testMethod"><xsl:value-of />
                </xsl:for-each></td>
             <xsl:apply-templates select="testMethod"/>
        </xsl:if>
    </tr>
</xsl:template>
<xsl:template match="teacher">
    <td><xsl:value-of select="@workUnit"/></td>
</xsl:template>
<xsl:template match="testMethod">
    <td><xsl:value-of select="@homeWork"/></td>
    <td><xsl:value-of select="lastHomeWork"/></td>
    <td><xsl:value-of select="test"/></td>
</xsl:template>
</xsl:stylesheet>
```

（4）将课后作业的次数大于 6 次以上的课程信息以某种形式的表格显示输出

例 8-11 中的代码示例是满足该功能要求的 Courses.xsl 文件，在输出的 HTML 标签中应用表格<table>标签以行、列的形式显示课程信息中的有关数据。此时例 8-8 中的 XML 文档示例代码在浏览器中的执行结果如图 8.4 所示。

地址(D) D:\schemaExample\Courses.xml 转到

下面为2010年软件学院部分课程信息表

课程名称	老师	老师的工作地	课时	上课教室	每周课次	介绍	测试方式	作业次数	最后的作业	测试
XML技术解析及应用	杨老师	中科院计算所	32学时	思源201	1	本课程是...	设计和实现XXXX系统 在课程结束进行考试	7	设计和实现XXXX系统	在课程结束进行考试
软件系统分析与设计技术实践	谢老师	中科院计算所	32学时	思源301	1	本课程是...	设计和实现XXXX系统 在课程结束进行考试	8	设计和实现XXXX系统	在课程结束进行考试
软件需求工程	谢老师	中科院计算所	32学时	思源401	1	本课程是...	设计和实现XXXX系统 在课程结束进行考试	8	设计和实现XXXX系统	在课程结束进行考试

图 8.4 例 8-8 中的 XML 文档示例代码在浏览器中的执行结果

例 8-11 查询课后作业的次数大于 6 次以上的课程信息的 XSLT 文件代码示例

```
<?xml version="1.0" encoding="gb2312" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
    <html><head><title>2010 年软件学院部分课程信息表</title></head>
        <body><xsl:apply-templates/></body>
    </html>
</xsl:template>
<xsl:template match=" Courses">
    <center><h3>下面为 2010 年软件学院部分课程信息表</h3></center>
    <table border="1">
        <th>课程名称</th><th>老师</th><th>老师的工作地</th><th>课时</th>
```

例 8-12　查询总课时为“48 学时”和“32 学时”的课程信息的 XSLT 文件代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/TR/WD - xsl">
<xsl:template match = "/">
    <html><head><title>2010 年软件学院部分课程信息表</title></head>
        <body><xsl:apply - templates/></body>
    </html>
</xsl:template>
<xsl:template match = " Courses">
    <center><h3>下面为 2010 年软件学院部分课程信息表</h3></center>
    <table border = "1">
        <th>课程名称</th><th>老师</th><th>课时</th><th>上课教室</th>
        <th>每周课次</th><th>介绍</th><th>测试方式</th><th>老师的工作地</th>
        <th>作业次数</th><th>最后的作业</th><th>测试</th>
        <xsl:apply - templates/>
    </table>
</xsl:template>
<xsl:template match = "Course">
    <xsl:choose>
       <tr><xsl:when test = ".[courseTimes  =  '48 学时']">
        <td><xsl:value - of select = "courseName"/></td>
        <td><xsl:for - each select = "teacher"><xsl:value - of /></xsl:for - each>
        </td>
        <td><xsl:value - of select = "courseTimes"/></td>
        <td><xsl:value - of select = "classRoom"/></td>
        <td><xsl:value - of select = "weekTimes"/></td>
        <td><xsl:value - of select = "instruction"/></td>
        <td><xsl:for - each select = "testMethod"><xsl:value - of />
             </xsl:for - each></td><xsl:apply - templates />
    </xsl:when>
    <xsl:when test = ".[courseTimes  =  '32 学时']">
        <td><xsl:value - of select = "courseName"/></td>
        <td><xsl:for - each select = "teacher"><xsl:value - of />
             </xsl:for - each></td>
        <td><xsl:value - of select = "courseTimes"/></td>
        <td><xsl:value - of select = "classRoom"/></td>
        <td><xsl:value - of select = "weekTimes"/></td>
        <td><xsl:value - of select = "instruction"/></td>
         <td><xsl:for - each select = "testMethod"><xsl:value - of />
             </xsl:for - each></td><xsl:apply - templates />
    </xsl:when>
   </tr>
    </xsl:choose>
</xsl:template>
<xsl:template match = "teacher">
    <td><xsl:value - of select = "@workUnit"/></td>
```

```
</xsl:template>
<xsl:template match = "testMethod">
    <td><xsl:value-of select = "@homeWork"/></td>
    <td><xsl:value-of select = "lastHomeWork"/></td>
    <td><xsl:value-of select = "test"/></td>
</xsl:template>
</xsl:stylesheet>
```

(6) 只显示输出 XML 文档中各个标签的注释内容，而其他的信息不需要输出

例 8-13 中的代码示例是满足该功能要求的 Courses.xsl 文件，在输出的 HTML 标签中应用表格<table>标签以行、列的形式显示课程信息中的有关注释数据。此时例 8-8 中的 XML 文档示例代码在浏览器中的执行结果如图 8.6 所示。

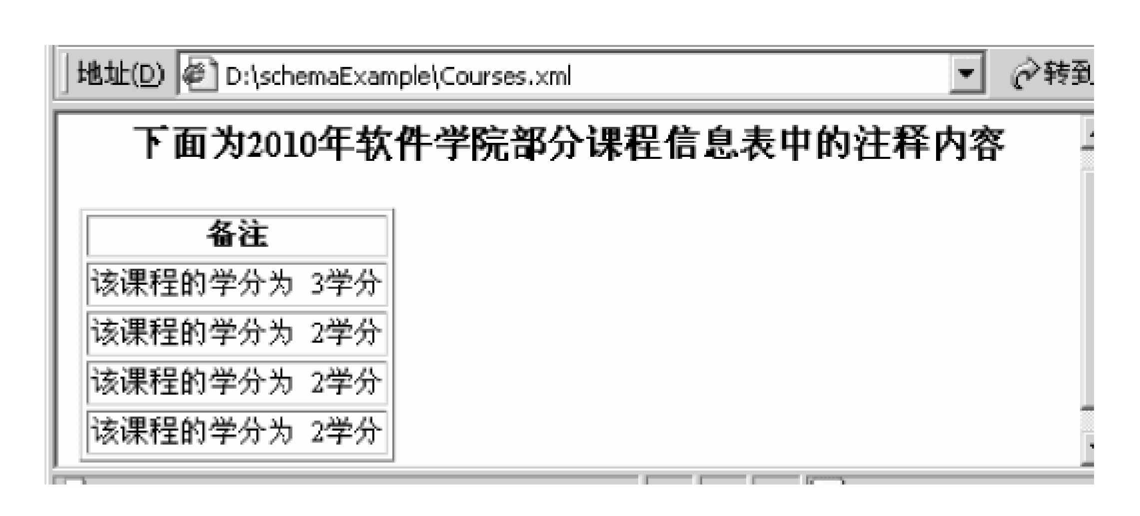


图 8.6　例 8-8 中的 XML 文档示例代码在浏览器中执行的结果

例 8-13　只显示输出 XML 文档中各个标签的注释内容的 XSLT 文件代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/TR/WD-xsl">
  <xsl:template match = "/">
    <html><head><title>2010 年软件学院部分课程信息表</title></head><body>
    <center><h3>下面为 2010 年软件学院部分课程信息表中的注释内容</h3></center>
    <table border = "1"><tr><th>备注</th></tr>
        <xsl:for-each select = "Courses/Course"><tr>
                <td><xsl:value-of select = "comment()"/></td></tr>
        </xsl:for-each></table>
    </body></html>
  </xsl:template>
</xsl:stylesheet>
```

8.2.3　XPath 中的轴及亲属关系匹配

1. XPath 中的轴及应用

XPath 中的轴(XPath Axes)可定义相对于当前节点的节点集，利用它可以实现亲属关系的节点匹配。因为 XML 文档中的各个节点形成树型结构，树中的任何一个节点都不是孤立的，节点之间的归属关系其实是一种亲属关系，如父、子、祖先、后代、兄弟等。当然，在对标签元素节点进行匹配时，可以充分应用节点之间所存在的这些亲属关系。

2. XPath 中的节点之间的各种关系及示例

(1) 父(Parent)节点

除根标签以外的其他标签元素节点及属性都允许有一个父节点。下面示例中的<testMethod>标签元素节点是<lastHomeWork>、属性“homeWork”及<test>的父节点。

```
<testMethod homeWork = "6">
    <lastHomeWork>利用所学习的 J2EE Web 技术设计和实现 XXXX 系统</lastHomeWork>
    <test>在课程结束进行考试</test>
</testMethod>
```

(2) 子(Children)节点

每个标签元素节点可有零个、一个或多个子节点。下面示例中的<lastHomeWork>节点及<test>节点是<testMethod>标签元素节点的子节点。

```
<testMethod homeWork = "6">
    <lastHomeWork>利用所学习的 J2EE Web 技术设计和实现 XXXX 系统</lastHomeWork>
    <test>在课程结束进行考试</test>
</testMethod>
```

(3) 同胞(Sibling)节点

拥有相同父节点的各个节点互为同胞(兄弟)节点。下面示例中的<lastHomeWork>、属性“homeWork”及<test>节点互为同胞(兄弟)节点。

```
<testMethod homeWork = "6">
    <lastHomeWork>利用所学习的 J2EE Web 技术设计和实现 XXXX 系统</lastHomeWork>
    <test>在课程结束进行考试</test>
</testMethod>
```

(4) 先辈(Ancestor)节点

泛指某个节点的父、祖父、祖父的父等更上级的各个层次的节点。下面示例中的<test>标签元素节点的先辈节点分别是<testMethod>和<oneCourseInfo>节点。

```
<oneCourseInfo>
     <testMethod homeWork = "6">
         <test>在课程结束进行考试</test>
      </testMethod>
</oneCourseInfo>
```

(5) 后代(Descendant)节点

某个节点的子节点、孙节点及孙节点的子节点等更低层次的节点。下面示例中的<oneCourseInfo>标签元素节点的后代节点分别是<testMethod>和<test>节点。

```
<oneCourseInfo>
     <testMethod homeWork = "6">
         <test>在课程结束进行考试</test>
      </testMethod>
</oneCourseInfo>
```

(10) preceding 轴

选取文档中当前节点的开始标签之前的所有节点,但不包括直接父节点、属性节点和命名空间节点。如示例:"/学生信息/专业/preceding:: * "表示<学生信息>元素到<专业>元素前面的所有节点元素,选择的结果为<姓名>和<出生日期>节点元素。

(11) preceding-sibling 轴

选取当前节点之前的所有同级节点。如示例:"/学生信息/出生日期/preceding-sibling:: * "表示<学生信息>元素到<出生日期>元素紧邻的前面所有兄弟节点元素,选择的结果为<姓名>元素。

(12) self 轴

选取当前节点元素本身。如示例:"//姓名/self:: * "表示所有<姓名>标签的自身元素,选择的结果为<姓名>标签元素本身。

4. XPath 中 child 轴的各种应用及示例

(1) child(子)轴的各种应用说明

- child::节点名称:选择所有属于当前节点的子元素中指定"节点名称"的节点。如示例:"child::学生信息"选取所有属于当前节点的子元素中的<学生信息>节点。
- child:: * :选取当前节点的所有子标签元素节点。如示例:"/child:: * "选取所有属于当前节点(<软件学院学生信息>标签元素节点)的子元素中的<学生信息>节点。
- child::text():选取当前节点的所有文本子节点。
- child::node():选取当前节点的第一个子节点的值。
- child::comment():选取当前节点的所有注释子节点。
- child::processing-instruction():选取所有处理器指令子节点。

(2) child 轴的应用示例

例 8-14 为一个体现 child 轴的技术特性的代码示例,其中以<软件学院学生信息>作为当前节点,并获得<软件学院学生信息>的所有子标签元素节点。例 8-14 中的 XSLT 示例文档所对应的 XML 文档在浏览器中预览的结果如图 8.7 所示。

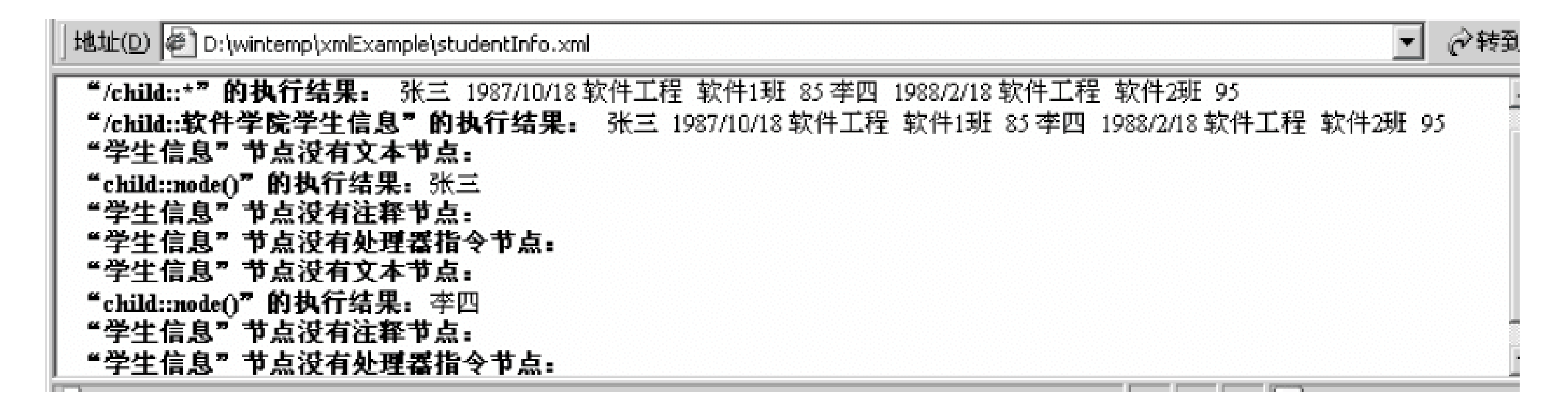


图 8.7　应用例 8-14 中的 XSLT 示例文档后的例 7-10 示例 XML 文档的执行结果

例 8-14　体现 child 轴的技术特性的代码示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" version = "1.0">
```

```
  <xsl:template match="/">
    <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
        <body><b>"/child:: * "的执行结果: </b>
            <xsl:value-of select="/child:: * "/><br/>
            <xsl:apply-templates select ="软件学院学生信息"/></body>
    </html>
</xsl:template>
<xsl:template match="软件学院学生信息">
      <b>"/child::软件学院学生信息"的执行结果: </b>
          <xsl:value-of select="/child::软件学院学生信息"/>
          <xsl:apply-templates select ="学生信息"/>
</xsl:template>
<xsl:template match="学生信息">
      <br/><b>"学生信息"节点没有文本节点: </b>
          <xsl:value-of select="child::text()"/>
      <br/><b>"child::node()"的执行结果: </b>
          <xsl:value-of select="child::node()"/>
      <br/><b>"学生信息"节点没有注释节点: </b>
          <xsl:value-of select="child::comment()"/>
      <br/><b>"学生信息"节点没有处理器指令节点: </b>
          <xsl:value-of select="child::processing-instruction()"/>
  </xsl:template>
</xsl:stylesheet>
```

5. XPath 中 descendant 轴的各种应用及示例

(1) descendant(后代)轴的各种应用说明

- descendant::节点名称：选取所有属于当前节点的后代元素中指定“节点名称”的节点。如示例：“descendant::学生信息”选取所有属于当前节点的后代元素中的<学生信息>节点。
- descendant:: * ：选取当前节点的所有后代标签元素节点。如示例：“/descendant:: * ”选取所有属于当前节点(<软件学院学生信息>标签元素节点)的后代元素中的<学生信息>节点。
- descendant::text()：选取当前节点的所有文本后代节点。
- descendant::node()：选取当前节点的第一个后代节点的值。
- descendant::comment()：选取当前节点的所有注释后代节点。
- descendant::processing—instruction()：选取所有处理器指令后代节点。

(2) descendant 轴的应用示例

例 8-15 为一个体现 descendant 轴的技术特性的代码示例，其中以<软件学院学生信息>作为当前节点，并获得<软件学院学生信息>的所有后代标签元素节点。例 8-15 中的 XSLT 示例文档所对应的 XML 文档在浏览器中预览的结果如图 8.8 所示。

2. XPath 节点集函数的应用示例

例 8-16 为一个体现 XPath 节点集函数功能的具体应用的代码示例，在该示例中统计 XML 课程平均成绩。注意其中黑体所标识的函数名。例 8-16 中的 XSLT 示例文档所对应的 XML 文档在浏览器中预览的结果如图 8.9 所示。

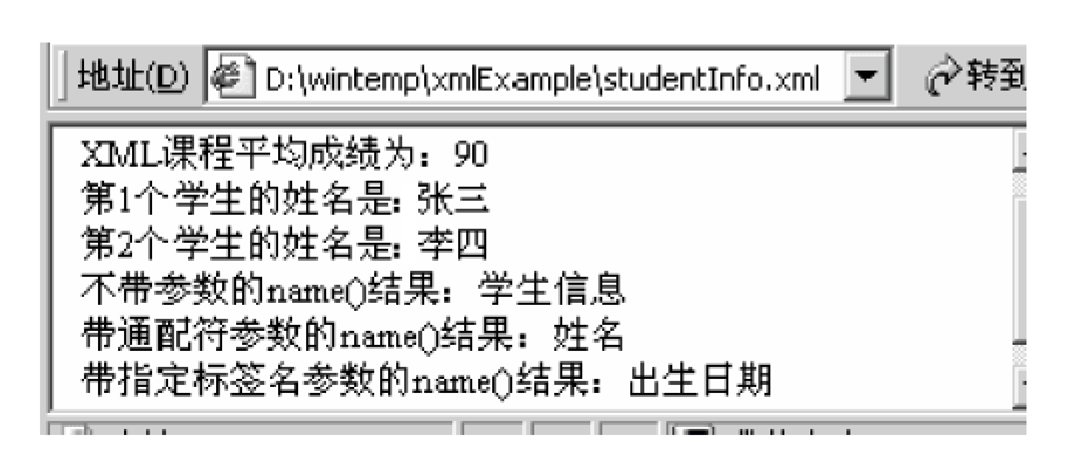


图 8.9 应用例 8-16 中的 XSLT 示例文档后的例 7-10 示例 XML 文档的执行结果

例 8-16 XPath 节点集函数的应用示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" version = "1.0">
  <xsl:template match = "/">
    <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
        <body>XML 课程平均成绩为:
            <xsl:value-of select = "sum(软件学院学生信息/学生信息/XML 考试成绩)
                                  div count(软件学院学生信息/学生信息)"/>
            <xsl:apply-templates select  = "软件学院学生信息"/>
        </body>
    </html>
</xsl:template>
<xsl:template match = "软件学院学生信息">
  <xsl:for-each select = "学生信息">
      <br/>第<xsl:value-of select = "position()"/>个学生的姓名是:
                                 <xsl:value-of select = "姓名"/>
  </xsl:for-each>
    <xsl:apply-templates select  = "学生信息[last()]"/>
  </xsl:template>
  <xsl:template match = "学生信息">
    <br/>不带参数的 name()结果: <xsl:value-of select = "name()"/>
    <br/>带通配符参数的 name()结果: <xsl:value-of select = "name( * )"/>
    <br/>带指定标签名参数的 name()结果: <xsl:value-of select = "name(出生日期)"/>
  </xsl:template>
</xsl:stylesheet>
```

8.3.2 XPath 中的字符串函数

1. string()字符串函数功能说明及转换规则

XPath 中的字符串函数实现对字符串进行各种形式的功能操作，其中的 string

(xpathExpression)函数实现将参数 xpathExpression(为其他类型的数据)的值转换为字符串,并按照如下的规则进行转换。

(1) 对于布尔类型、节点类型参数的转换规则

- 如果参数为布尔类型的 true,将返回 true 字符串;如果参数为布尔类型的 false,将返回 false 字符串。
- 对于节点类型的参数,将返回该节点内各个子节点所对应的数据值;如果节点集为空,则返回一个空字符串。

(2) 对于数值类型的参数转换规则

- NaN(Not a Number,不是一个有效的数字):转换为字符串 NaN。
- 正零:转换为字符串 0。
- 负零:转换为字符串 0。
- 正无穷大:转换为 Infinity。
- 负无穷大:转换为-Infinity。
- 如果参数为整数,则被表示为没有小数点和前导零的十进制格式的数字。如果参数为负整数,则在它的前面加一个负号(—)。
- 如果参数为带小数点的实数,数值被表示为一个包含有小数点(小数点前后至少各有一位数字)的十进制格式的数字。如果数字是负数,则在它的前面加一个符号(—)。

2. XPath 中的主要字符串函数功能说明及应用要点

(1) string concat(string,string,string *)函数

将多个字符串进行连接,并返回连接后的结果字符串。

(2) boolean contains(string,string)函数

判断某个字符串是否是另一个字符串中的一部分,如果第一个字符串包含第二个字符串,则返回真,否则返回假。

如示例:<xsl:value-of select="// * [contains(name(),'学生')]"/>,表示选择所有元素名称中包含有“学生”字符串内容的标签元素节点。

(3) string normalize-space(string?)函数

删除字符串的头部和尾部的空白字符,并返回除掉了前后空白字符及重复空白的参数字符串(如果字符串中间含有多个连续的空白字符,将用一个空格来代替)。如果没有给定具体的参数,将自动对当前节点进行转换。

如示例:<xsl:value-of select="//学生信息[normalize-space(@性别)='男']"/>,选择含有“性别”属性且其值(在用 normalize-space()函数除掉前后空格后的属性值)为“男”的学生信息节点。

(4) boolean start-with(stringOne,stringTwo)函数

用来判断 stringOne 是否以 stringTwo 开头,如果满足这样的条件,则返回 true,否则返回 false。

如示例:<xsl:value-of select="// * [starts-with(name(),'学生')]"/>,表示选择所有标签名称以“学生”字符串作为起始的标签元素节点。

例 8-17 XPath 字符串函数的应用示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"version = "1.0">
  <xsl:template match = "/">
    <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
        <body><xsl:apply-templates select  = "软件学院学生信息"/></body>
    </html>
</xsl:template>
<xsl:template match = "软件学院学生信息">
        <xsl:apply-templates select  = "学生信息[1]"/>
</xsl:template>
<xsl:template match = "学生信息[1]">
      <b>concat 函数的执行结果: </b><xsl:value-of select = "concat(姓名,@性别,
                                出生日期,专业,班级,@方向,XML 考试成绩)" /><br/>
      <b>contains 函数的执行结果: </b>
          <xsl:value-of select = "//*[contains(name(),'学生')]" /><br/>
      <b>normalize-space 函数的执行结果: </b>
      <xsl:value-of select = "//学生信息[normalize-space(@性别) = '男']"/><br/>
      <b>starts-with 函数的执行结果: </b>
      <xsl:value-of select = "//*[starts-with(name(),'学生')]" /><br/>
      <b>string-length 函数的执行结果: </b>
          <xsl:value-of select = "string-length(姓名)" /><br/>
      <b>substring 函数的执行结果: </b>
          <xsl:value-of select = "substring(出生日期, 1, 4)" /><br/>
      <b>substring-before 函数的执行结果: </b>
          <xsl:value-of select = "substring-before(出生日期, '/')" /><br/>
      <b>substring-after 函数的执行结果: </b>
          <xsl:value-of select = "substring-after(出生日期, '/')" /><br/>
      <b>translate 函数的执行结果: </b>
          <xsl:value-of select = "translate(班级, '软件','软工')" /><br/>
      <b>string 函数对节点类型转换的结果: </b>
          <xsl:value-of select = "string(.)" /><br/>
      <b>string 函数对布尔数据转换的结果: </b>
          <xsl:value-of select = "string(XML 考试成绩> 96)" /><br/>
  </xsl:template>
</xsl:stylesheet>
```

8.3.3 XPath 中的布尔函数

1. boolean()函数功能说明

boolean()(布尔)函数主要用于对给定的参数表达式进行计算,并返回布尔类型的结果值。其中的 boolean(xpathExpression)函数用于将参数 xpathExpression 的值转换为布尔类型,因此可以实现将其他类型的数据转换为布尔类型。如果参数为数值 0,boolean()函数将返回 false,其他数值的数字类型的参数都返回 true;对于节点类型的参数,如果节点存在

(2) number floor(digit)函数

返回小于或等于数值 digit 的最大整数。当给定的参数无法转换为数值类型的值时,将返回 NaN。

(3) number number(表达式参数)函数

用于将"表达式参数"的参数值转换为数值。如示例:<xsl:value-of select="姓名[number(studentAge)<25]"/>,获得年龄小于 25 岁的学生姓名,其中的 number()函数实现将节点值中的文本转换为数值。

如果没有给 number()函数提供参数,将自动应用当前节点作为参数;当给定的参数无法转换为数值类型的值时,将返回 NaN。而如果给定的参数为布尔类型的 true 值时,将返回结果值 1;布尔类型的参数值为 false 时,则返回结果值 0。

(4) number round(表达式参数)函数

用于将"表达式参数"的参数值进行四舍五入取整,其中的"表达式参数"是必选项,如果所给定的参数是无法转换为数值类型的参数值时,将返回 NaN。而如果给定的参数为布尔类型的 true 值时,将返回结果值 1;布尔类型的参数值为 false 时,则返回结果值 0。

(5) number sum(节点集)函数

返回指定节点集中所有节点值的总和值,每个节点在求和之前都要先转换为数值。如示例:本学期 XML 课程的考试总成绩=<xsl:value-of select="sum(//testScore)"/>,计算学生某个课程考试的总成绩值。

2. XPath 中的数值函数应用示例

例 8-19 为一个体现 XPath 数值函数功能的具体应用的代码示例,在该示例中应用 XPath 中的各个数值函数,并显示各个函数的执行结果。例 8-19 中的 XSLT 示例文档所对应的 XML 文档在浏览器中预览的结果如图 8.11 所示。

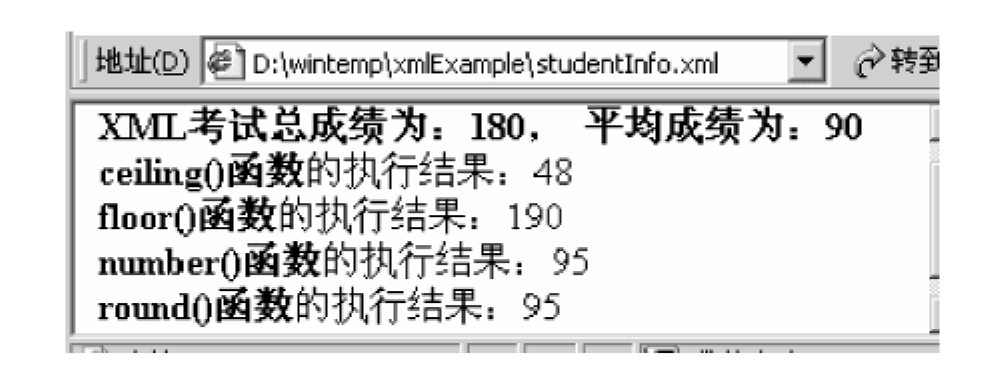


图 8.11 应用例 8-19 中的 XSLT 示例文档后的例 7-10 示例 XML 文档的执行结果

例 8-19 XPath 数值函数的应用示例

```
<?xml version = "1.0" encoding = "gb2312" ?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"version = "1.0">
  <xsl:template match = "/">
    <html><head><title>北京蓝梦大学软件学院学生信息</title></head>
        <body><xsl:apply-templates select = "软件学院学生信息"/></body>
    </html>
  </xsl:template>
```

- XPathFunctionResolver：提供了对用户定义的 XPathFunction 函数集的访问。
- XPathVariableResolver：提供了对用户定义的 XPath 变量集的访问。

XPath 对象不是线程安全的，也不能重复载入。也就是说，应用程序负责确保在任意给定时间内不能有多个线程使用一个 XPath 对象。

XPathExpression 接口提供了对编译后的 XPath 表达式的访问。如果表达式包含变量，则其值将通过 XPathVariableResolver()方法找到；如果表达式包含函数引用，则函数将通过 XPathFunctionResolver()方法找到。

XPathFunction 接口提供了对 XPath 函数的访问，这个接口只声明了一个如下的方法 evaluate：public Object evaluate(List args)throws XPathFunctionException。

而 XPathVariableResolver 接口和 XPathFunctionResolver 接口主要用于在程序中定义和使用 XPath 的扩展函数(用户自定义的函数)，一般不常使用。

(2) XPathFactory 类

XPathFactory 是一个抽象工厂类，主要用于创建 XPath 对象。在该类中只有一个受保护的空的构造方法。之所以要设计为抽象工厂，主要的设计目标是希望能够支持不同的对象模型，如 DOM、JDOM 和 XOM 等。为了能够选择不同的对象模型，只需要向 XPathFactory 类中的静态方法 newInstance()传递标识对象模型的统一资源标识符(URI)即可。但一般只需要应用不带参数的静态方法 newInstance()，用于获取使用默认对象模型的 XPathFactory 的对象实例。

3. XPath 的数据类型在 Java 中的对应关系

XPath 和 Java 语言彼此之间没有相同的数据类型系统，在 XPath 1.0 中只有节点集、布尔值、字符串和数值 4 种基本的数据类型，但在 Java 语言中提供了丰富的数据类型，包括集合和用户自定义的类型。因此，需要在 XPath 和 Java 之间对数据类型进行相互转换。

其中，将 XPath 中的节点集映射到 Java 中的 org.w3c.dom.NodeList 类，而将 XPath 中的数值类型映射到 Java 中的 java.lang.Double 类，将 XPath 中的布尔类型映射到 Java 中的 java.lang.Boolean 类，将 XPath 中的字符串类型映射到 Java 中的 java.lang.String 字符串类。

8.4.2 在 Java 程序中应用 Java XPath API

1. 应用 Java XPath API 的主要编程方法

(1) 将 XML 文档加载到 DOM Document 对象中

可以应用 javax.xml.parsers.DocumentBuilderFactory 类中的静态方法 newInstance()获得一个平台默认的 DocumentBuilderFactory 的新实例；如何应用 DocumentBuilderFactory 类中的 newDocumentBuilder()方法获得一个 DocumentBuilder 类的对象实例，参见如下代码片段示例：

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
```

XPathConstants. NODE,则 evaluate()方法将返回 null。

(6) 将返回结果强制转换成 DOM 的 NodeList 类型对象

通过将结果强制转换成 Java DOM API 中的 NodeList 类型对象,就可以遍历节点集中的各个节点对象,获得目标节点数据。参见如下代码片段示例:

```
NodeList nodes = (NodeList) result;
for (int i = 0; i < nodes.getLength(); i++){
    System.out.println(nodes.item(i).getNodeValue());
}
```

2. 应用 Java XPath API 的程序示例

本示例操作的目标 XML 文档仍然以例 7-10 中描述软件学院学生信息的 XML 文档为示例。例 8-20 中的示例代码实现在 Java 程序中利用 XPath API 操作例 7-10 中的 XML 文档,文件名称设为“softWareStudentInfo. xml”。

在例 8-20 示例程序中设计了 3 个功能方法。其中的 createDOMDocument()方法动态加载例 7-10 示例中的 XML 文档,并获得一个代表该文档的 DOM 树模型 Document 对象;returnXPathResult()方法根据给定的 XPath 节点定位的表达式在 XML 文档中获得目标节点集对象;最后利用 dealResultNodes()方法处理目标节点集中所需要的节点数据。本示例只是简单地获得学生信息中女生的姓名。

例 8-20 应用 Java XPath API 的代码示例

```
package com.px1987.webcrm.xpath;
import java.io.IOException;
import java.io.InputStream;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
public class JavaXPathDemo {
    public JavaXPathDemo() {
    }
    public Document createDOMDocument() throws ParserConfigurationException,
                                    SAXException, IOException{
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        InputStream xmlInputStream =
```

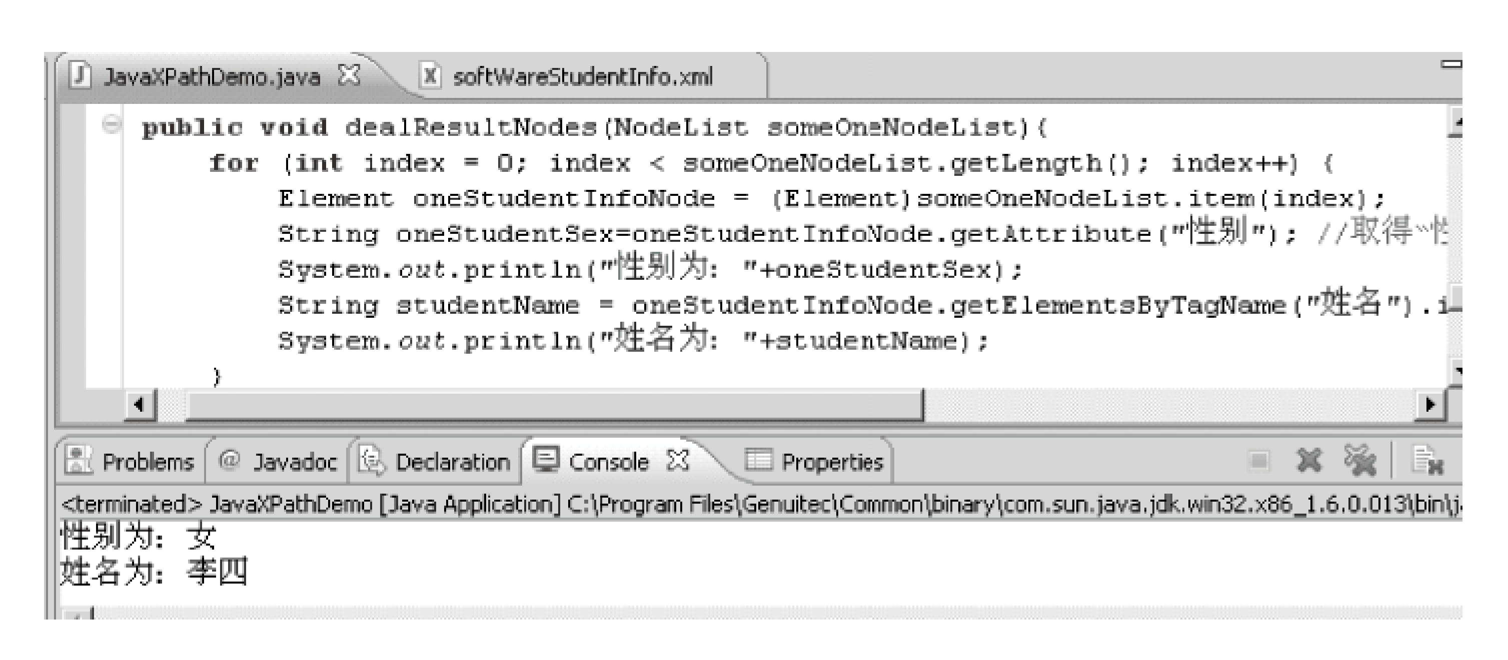


图 8.12　例 8-20 示例的最终执行结果

本章小结

教学重点

XPath 是一种在 XML 文档中查找目标节点信息的查询语言，它可遍历 XML 文档中的节点元素和属性，同时也是 W3C XSLT 标准的主要组成部分，并且 XQuery 和 XPointer 等技术也构建于 XPath 基础之上。因此，熟练地掌握 XPath 和正确地应用 XPath 及有关的技术是深入学习和应用 XML 的技术基础。

本章的主要教学重点首先是对 XML XPath 技术的了解及掌握它的主要应用，包括 XPath 技术的主要特性及语法规则、XPath 中各种形式的操作符及具体的应用、XPath 中的功能函数及应用等方面的内容；其次是熟悉和掌握 Java 平台对 XPath 技术的支持，包括 javax. xml. xpath 软件包中的核心功能类和接口的编程应用。

学习难点

在 XPath 中可以采用路径匹配、位置匹配、属性及属性值匹配、亲属关系匹配和条件匹配等形式进行节点定位。而其中的亲属关系匹配就是利用节点之间的层次关系和归属关系定位目标节点，也就不可避免地要涉及和应用 XPath 中的轴。

对于 XPath 中的轴，可以理解为以某个节点为中心，向四周(上/下表现为先辈和后代节点、左/右则表现为同辈兄弟节点)导航定位的坐标系。

教学要点

在使用 XML 文档中的数据时，常见的功能操作要求是查找给定节点的值。而通过使用 XPath 可以简化对 XML 文档中目标数据的遍历过程，为此在 XPath 中对 XML 文档重新分类，共提供了 7 种不同类型的节点。在教学中需要向学生理顺这些不同节点的内涵和区别。

2. 填空题

（1）XPath 中的 7 种主要节点类型分别是________、________、________、________、________、________和________。

（2）XPath 中的 4 种数据类型分别是________、________、________和________。

（3）在 XPath 中提供了轴定位导航 XML 文档中的目标节点，其中 ancestor 轴的定位规则是________，attribute 轴的定位规则是________，child 轴的定位规则是________，descendant 轴的定位规则是________。

（4）XPath 中主要有 4 类函数，它们分别是________、________、________和________。

（5）XPath 中的逻辑运算符号主要有________、________，关系运算符号主要有________、________、________、________等。

3. 问答题

（1）什么是 XPath？它的主要作用体现在哪些方面？为什么在应用 XSLT 时，也要应用 XPath？

（2）通过具体的 XML 文档示例说明 XPath 中的 7 种主要节点类型的含义。在 XPath 中有哪些数据类型？

（3）在 XPath 中有哪些节点定位方式？什么是路径匹配？什么是位置匹配？

（4）通过具体的示例说明属性及属性值匹配、亲属关系匹配和条件匹配等方式的具体应用。

（5）什么是 XPath 中的轴？通过具体的示例说明 XPath 中几种典型的轴在节点定位中的应用。

4. 开发题

（1）以例 8-1 示例中的描述 J2EE 架构与程序设计必修课信息的 XML 文档为目标 XML 文档，利用 Java XPath API 编写一个访问该 XML 文档中数据的 Java 程序。功能要求如下：

- 查询授课老师为“杨老师”所讲的各个课程信息；
- 查询课后作业的次数多于 4 次以上的课程信息；
- 查询总课时为“48 学时”的课程信息。

（2）以例 7-10 示例中的描述软件学院学生信息的 XML 文档为目标 XML 文档，按照下面的功能要求设计对应的 XSLT 文件。

- 查询性别为“男性”的所有学生信息；
- 查询专业方向为“J2EE 软件开发”的所有学生信息；
- 查询 XML 考试成绩高于 85 分的所有学生信息。

于 1998 年 10 月 1 日成为 W3C 推荐的标准。

2. W3C DOM 有三个不同的级别

DOM 规范在发展和完善的过程中，有三个不同层次和级别的标准。目前多数 Web 浏览器都支持和实现了 DOM Level 2(DOM 2 级)规范中的多数接口。

(1) DOM Level 1 规范

DOM Level 1 规范的重点在于解决了 Netscape 网景公司的 JavaScript 脚本语言和微软的 JScript 脚本语言之间的冲突，为 Web 开发人员提供一个标准和统一的访问 HTML 页面内各个标签元素的编程接口。这也是 HTML DOM 产生的技术背景。因此，HTML DOM 是一种与浏览器、运行的操作系统平台和脚本语言无关的应用程序编程接口。

(2) DOM Level 2 规范

尽管 DOM Level 1 规范提供了包括对 XML1.0 和 HTML 的支持，但由于没有包括对 XML 命名空间的支持，而且也只提供了对 HTML 页面中核心节点的模型化描述。因此，在应用方面基本上还只局限于利用 JavaScript 等 Web 脚本语言动态操作 HTML 标签元素，并没有真正地应用在对 XML 文档的数据解析方面。微软和网景公司在此基础上，又分别各自扩展 W3C 的 HTML DOM 规范形成自己的 DHTML 技术。

DOM Level 2 规范不仅添加了对 XML 命名空间的支持，而且还扩展了 DOM Level 1 规范，增加了对文档创建、样式表对象模型和 HTML 元素的事件模型的支持，W3C 最终于 2000 年 11 月 13 日正式发布 DOM Level 2 规范。DOM Level 2 规范的发布为应用程序动态操作 XML 文档中的各种元素提供了一个统一的、与平台无关的编程接口。

(3) DOM Level 3 规范

W3C 的 DOM Level 3 规范增加了对 DTD 和 Schemas 等内容模型的技术支持，并提供了文档验证的统一标准接口，同时还进一步规定了对与 HTML/XML 文档加载和保存、文档查看、文档格式化等有关的各种事件模型的支持。当然，DOM Level 3 规范同样也建立于 DOM Level 2 规范基础之上，W3C 于 2004 年 4 月 7 日正式发布了 DOM Level 3 规范。

随着 W3C 不断地对 DOM 技术及规范的完善和升级，基于 DOM 技术实现的各种 XML 解析程序和应用系统也层出不穷。读者可以浏览 W3C 关于 DOM 技术介绍的官方网站 http://www.w3.org/DOM/，详细地了解 DOM 不同级别的核心内容和差别，如图 9.1 所示。

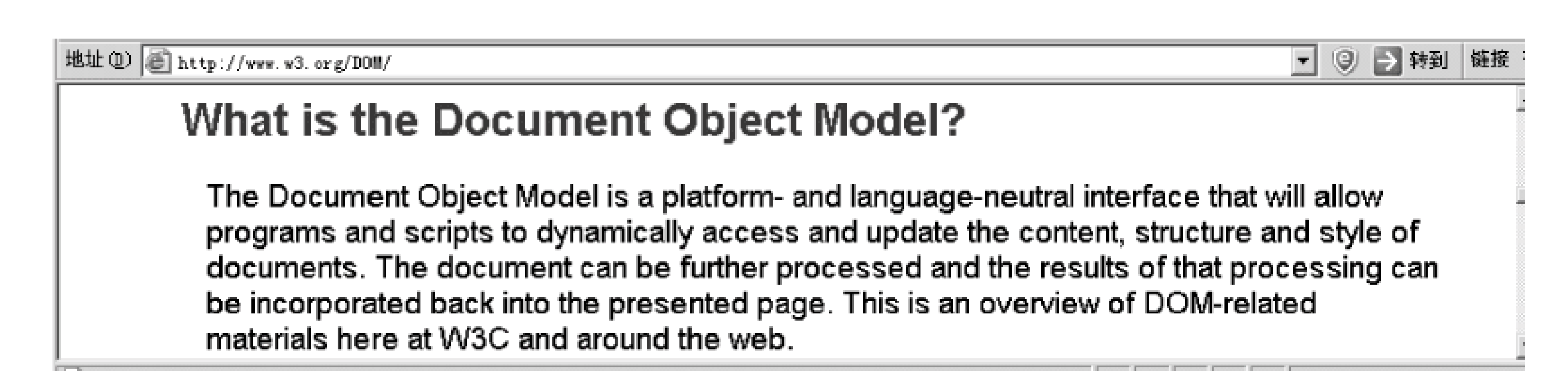


图 9.1　W3C 的官方网站中关于 DOM 技术介绍的信息

在某种编程语言环境中应用 DOM API 编程操作 HTML/XML 文档中的数据时，可以利用 DOMImplementation 接口中的 boolean hasFeature(String feature,String version)方

素，可以包含处理器指令节点、注释节点、文本节点、CDATA 段节点和实体引用节点作为子节点。

- 实体节点(Entity Node)：在 DTD 中实体定义的内容，可以包含处理器指令节点、注释节点、文本节点、CDATA 段节点和实体引用节点作为子节点。
- 注释节点(Comment Node)：HTML/XML 文档中的注释文本，它没有子节点。
- 文档片段节点(Document Fragment Node)：对 XML 文档中相对独立部分的标签元素集的包装，其中可以包含处理器指令节点、注释节点、文本节点、CDATA 段节点和实体引用节点作为子节点。
- Notation 节点(Notation Node)：在 DTD 中声明的外部符号，它没有子节点。

3. W3C 在 DOM API 规范中不同节点类型的数值化表示

为了方便开发人员在程序中识别 DOM 树模型中某个节点的类型，W3C 在 DOM API 规范中为不同的节点类型提供对应的数值化表示和在 Node 接口中定义的命名符号常量，如表 9.1 所示。在程序中利用 Node 接口中的 nodeType 成员属性可以获得当前节点类型的数值，从而可以识别当前节点的类型。

表 9.1　不同的节点类型所对应的数值化表示和命名的符号常量

节 点 类 型	节点类型的数值	符 号 常 量	DOM API 中的接口名
标签元素节点	1	ELEMENT_NODE	Element
属性节点	2	ATTRIBUTE_NODE	Attr
文本节点	3	TEXT_NODE	Text
CDATA 段节点	4	CDATA_SECTION_NODE	CDATASection
实体引用节点	5	ENTITY_REFERENCE_NODE	EntityReference
实体节点	6	ENTITY_NODE	Entity
处理器指令节点	7	PROCESSING_INSTRUCTION_NODE	ProcessingInstruction
注释节点	8	COMMENT_NODE	Comment
文档节点	9	DOCUMENT_NODE	Document
文档类型节点	10	DOCUMENT_TYPE_NODE	DocumentType
文档片段节点	11	DOCUMENT_FRAGMENT_NODE	DocumentFragment
Notation 节点	12	NOTATION_NODE	Notation

下面的 JavaScript 代码片段示例识别当前节点对象 someOneNode 是否为标签元素节点对象。

```
if(someOneNode.nodeType == Node.ELEMENT_NODE){
  //表明当前节点为标签元素对象节点
}
```

这段代码可以在 Mozilla 1.0+、Opera 7.0+和 Safari 1.0+等浏览器中正常运行。但由于微软 IE 浏览器不支持 Node 接口符号名和这些命名符号常量，在 IE 中执行会产生错误。需要改用数值判断，如下为适应微软 IE 浏览器的 JavaScript 代码片段示例。

```
if(someOneNode.nodeType == 1){
```

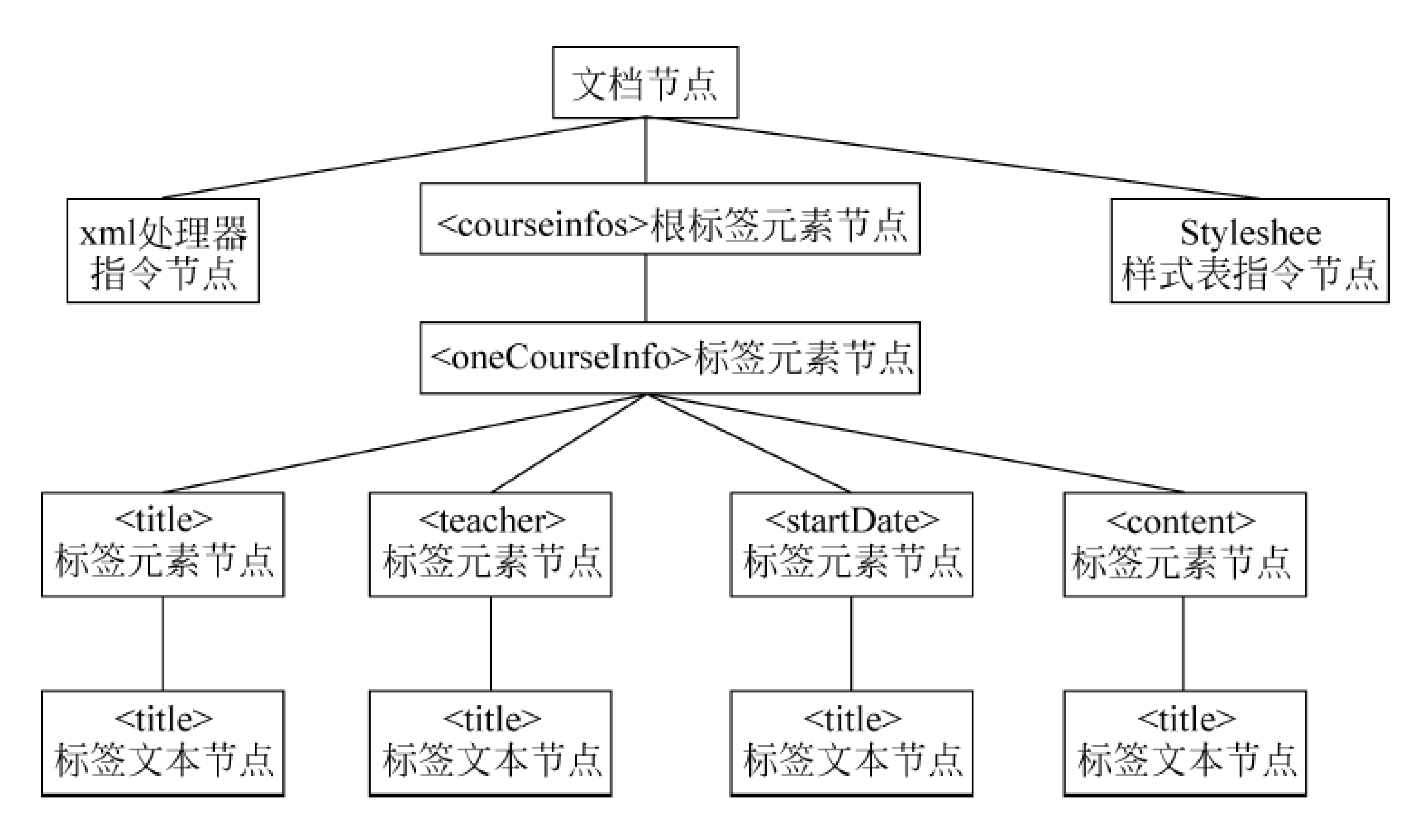


图 9.2 例 7-8 示例所对应的 XML DOM 树模型中的各个节点的逻辑结构图

在应用中需要区分标签元素节点和文本节点之间的差别，尽管文本节点可以作为标签元素节点的子节点，但文本节点并不是标签元素节点的值。如例 7-8 示例中的＜teacher＞张老师＜/teacher＞标签，其中的＜teacher＞标签属于标签元素节点，其值为“teacher”；而“张老师”被抽象为文本节点，该文本节点的值为“张老师”。

6. HTML DOM 树模型内部的逻辑结构

(1) HTML DOM 树模型和 XML DOM 树模型采用一致的树形结构表示

HTML 文档中的每个标签元素、属性、文本等同样都在 DOM 树模型中有对应的对象表示，DOM 树模型起始于 HTML 文档节点，并按照 HTML 文档中各个元素之间的关系形成节点树中的各个节点集。如图 9.3 所示为例 1-1 所示的 HTML 页面示例所对应的 HTML DOM 树模型中各个节点的逻辑结构图。

HTML 文档的 DOM 树模型和 XML 文档的 DOM 树模型采用一致的树形结构表示，因此，可以采用与解析 XML 文档一致的程序设计风格解析 HTML 文档中的各个节点元素。但其中的＜html＞标签尽管是 HTML 文档的根标签，但在 DOM 树模型中却是文档(Domcument)节点的子节点，而＜head＞和＜body＞都是＜html＞的子节点，并且彼此之间互为兄弟节点。

(2) DOM 树模型中的节点与文档中的元素不是完全等价的

DOM 树模型中的节点和 HTML/XML 文档中的元素并不完全等价，文档中的元素是指一对标签符(Tag)及其内部包含的字符串值的总和。例如，下面的＜title＞标签只是一个元素：＜title＞这是我的 HelloWorld 页面＜/title＞，但它并不是一个节点，而是两个。其中第一个节点是＜title＞标签元素(Element)节点，它的值是 null；第二个节点是一个文本(Text)节点，它的值是“这是我的 HelloWorld 页面\n”，该文本节点是＜title＞标签元素节点的子节点。

同样，标签元素的属性(Attr)节点也不属于对应的标签元素的子节点，如图 9.3 所示的＜a＞标签 href 属性节点并不是＜a＞标签元素节点的子节点(黑体所标识的内容)，因为属

为此，在 Java 平台中开源的 JDom 系统尽管在设计思想上仍然基于 DOM 树模型，但专门针对 Java 技术平台而设计。因此，在 JDom 中充分应用了 Java 平台中的许多技术特性，如多态、反射和序列化等核心技术改进 DOM 本身所存在的缺陷。

2. DOM API 中的接口主要分为基本接口和扩展接口

基本接口（Fundamental Interfaces）是 DOM API 规范中其他接口的基础，定义了针对 HTML/XML 文档中都具有的节点对象编程接口，如 Document、Node、Element、Attr、Text 和 Comment 等接口；而在扩展接口（Extended Interfaces）中提供了专门针对 XML 文档中节点对象的编程接口，如 CDATASection、EntityReference、Entity、Notation、ProcessingInstruction 和 DocumentType 等接口。

3. DOM API 中的 Node 接口

在 DOM API 规范中提供了两套不同形式的接口来操作 HTML/XML 文档数据，其中一套是以面向对象的方式将 DOM 树模型中的不同节点对象组织成具有继承关系的层次树结构，并按照树中的目标节点对象操作 HTML/XML 文档中对应的元素；而另一套接口则以节点对象（Node）为核心，对 HTML/XML 文档中的所有元素操作都通过 Node 接口中的方法来实现。

（1）Node 接口中的各个子接口

由于 Node（节点）是一个相对抽象的概念，因此 Node 接口被设计为抽象的接口，它是对 HTML/XML 文档树中的一个独立元素的抽象描述，并在该 Node 接口的基础上扩展了如下几个主要的子接口。

- Document：表示整个 HTML 或 XML 文档。
- Element：表示 HTML/XML 文档中的某个标签元素节点对象。
- Attr：表示 HTML/XML 文档中的某个标签元素的属性节点对象。
- Text：表示 HTML/XML 文档中某个标签元素节点对象中的文本节点对象。
- Comment：表示 HTML/XML 文档中的注释文本所对应的节点对象。
- CDATASection：表示 XML 文档中的某个 CDATA 段内容所对应的节点对象。

正是由于 Node 接口是一个相对抽象的概念，在实际编程中一般都不会真正地直接应用 Node 接口本身，而是使用它的各个子接口所对应的对象操作 HTML/XML 文档。

（2）Node 接口中所包含的主要方法定义及功能

- appendChild()：通过把一个节点对象添加到当前节点对象的 childNodes[]数组中，实现在文档树中增加某个节点对象。
- cloneNode()：复制当前节点对象，或者复制当前节点及它的所有后代子孙节点。
- hasChildNodes()：识别当前节点是否拥有子节点，返回 true 表示有子节点。
- insertBefore()：在文档树中插入一个节点对象，插入的位置是在当前节点的指定子节点之前。如果该节点已经存在，则删除该节点后再插入到它的位置。
- removeChild()：删除指定节点对象的子节点对象，并返回删除后的子节点对象。
- replaceChild()：用另一个节点替换指定节点对象的子节点对象。

- createElement()：用指定的标签名新建标签元素节点。
- createTextNode()：用指定的文本字符串创建文本节点。
- getElementById()：获得当前文档中指定 id 属性值的标签元素节点对象。
- getElementsByTagName()：获得当前文档中指定标签名的所有的标签元素节点对象，由于有可能存在多个同名的标签元素，返回值为节点集 NodeList 对象。
- getDocumentElement()：获得文档中根标签元素的 Element 对象。

因此，无论创建什么类型的 Node 节点对象，都可以通过 Document 对象中的 createXXX()方法实现，其中的 XXX 代表具体要创建的节点类型名。比如创建标签元素 Element 对象，可以应用 createElement()方法。

5. DOM API 中 Node 接口的子接口 Element 接口

(1) Element 接口的主要作用

它代表 HTML/XML 文档中某一具体的标签元素，在标签元素中可以包含属性、其他标签元素(形成子标签元素)或文本。如果标签元素中含有文本内容，则该文本内容被包装到文本对象中。同样，标签元素中的属性也被包装到属性对象(Attr 接口)中。

(2) Element 接口中常用的方法定义及功能

- getAttribute()：以字符串形式返回指定名称的属性值。
- getAttributeNode()：以 Attr 节点对象的形式返回指定名称的属性对象。
- getElementsByTagName()：返回指定标签名的 NodeList 集合对象，其中包含了所有的子孙节点对象，但这些子孙节点对象的顺序与在文档中出现的顺序要保持一致。
- hasAttribute()：识别该元素对象是否有指定名字的属性存在，返回 true 表示存在特定名称的属性。
- removeAttribute()：从当前元素对象中删除指定名称的属性。
- removeAttributeNode()：从当前元素对象的属性集中删除指定名称的 Attr 对象。
- setAttribute()：为指定名称的属性设置指定的字符串值，如果该属性不存在，则创建该属性。
- setAttributeNode()：把指定的 Attr 节点对象添加到该元素对象的属性集中。

(3) 如何修改或获得某个标签元素节点的正文内容

要修改某个标签元素节点的正文内容，首先要定位该标签元素的节点，这可以用 Element 接口中的 getElementsByTagName(String name)方法获得所有代表该标签元素的节点对象；然后就可以用 Node 接口中的 getFirstChild()方法得到代表该标签元素正文内容的文本节点对象，用 Node 接口中的 setNodeValue(String)方法可以修改文本节点对象的值，或者用 getNodeValue()方法获得文本节点对象的值。用 Java 实现的代码片段示例如下：

```
Node oneNodeObj = someOneElement.getElementsByTagName("tagName").item(0);
String someOneElementText = oneNodeObj.getFirstChild().getNodeValue();
```

在编程中需要注意的一点是，对标签元素节点的正文内容不能直接用 Element 对象的

```
            domDocument.createProcessingInstruction("xml - stylesheet",
                                  "type = \"text/css\" href = \"studentInfo.css\"");
domDocument.appendChild(pi);
```

由于 XML 解析程序本身并不对给定的处理器指令的内容字符串进行任何的词汇检查，因此如果在该内容中存在字符序列“<?”和“?>”，将会在输出 XML 文档时产生异常。

9. DOM API 中的 NodeList 接口

NodeList 对象代表一组有序的 Node 节点对象集合，通过对该集合进行迭代，可以获得其中的各个 Node 节点对象。在 NodeList 接口中提供了 getLength()方法返回节点对象集合的长度(对象的个数)，item(int)方法返回指定位置序号的 Node 节点对象。

10. DOM API 中的 Comment 接口和 Text 接口

由于 Text 节点对象和 Comment 节点对象都是纯文本，因此在 DOM API 中的 Comment 接口和 Text 接口都继承了一个共同的 CharacterData(文本字符)接口。

为了获得 Text 节点对象或者 Comment 节点对象中的文本内容，可以应用 CharacterData 接口中的 data 成员属性，或者使用 Node 接口中继承的 nodevalue 成员属性，或者使用 getNodeValue()成员方法。

9.2 Java 中的 DOM 技术实现

9.2.1 Java 平台中的 JAXP API

1. Sun 公司的 Java XML 解析系统库 JAXP API

(1) Sun 公司在 Java 技术平台中提供了解析 XML 文档的 JAXP API

JAXP API(Java API for XML Processing)的中文含义为：用于 XML 文档处理的使用 Java 语言编写的编程接口。它支持 DOM、SAX、XSLT 等标准，并提供了对 XML 文档中的各个元素对象实现基本的读、写、增、删、改和查询等功能，通过它可以很方便地将 XML 文档集成到 Java 平台的各种应用程序中。

(2) JAXP 具有可插拔性(Pluggability)

JAXP 的精华所在是它的“可插拔性”，也就是 XML 应用开发人员可以编写自己的 XML 解析处理程序，只要它符合 JAXP 的 APIs，就可以实现与底层的 XML 处理器平台无关性。因为 JAXP 只是一个 API 规范，而真正底层的实现平台是可以允许多种形式的。

(3) JAXP 的两种实现方式主要为 DOM2 API 和 SAX2 API

JAXP 支持基于 DOM 树模型对象和基于 SAX 事件流两种解析方式，在 J2SDK 1.4 以上版的系统库中，org.w3c.dom 软件包中提供了对 W3C DOM Level 2 Core 推荐接口的 Java 平台实现，org.w3c.dom 软件包的各个类和接口打包在 rt.jar 文件中，该文件在 J2SDK 的安装目录 C:\Program Files\Java\jdk1.6.0\jre\lib 中，如图 9.4 所示。

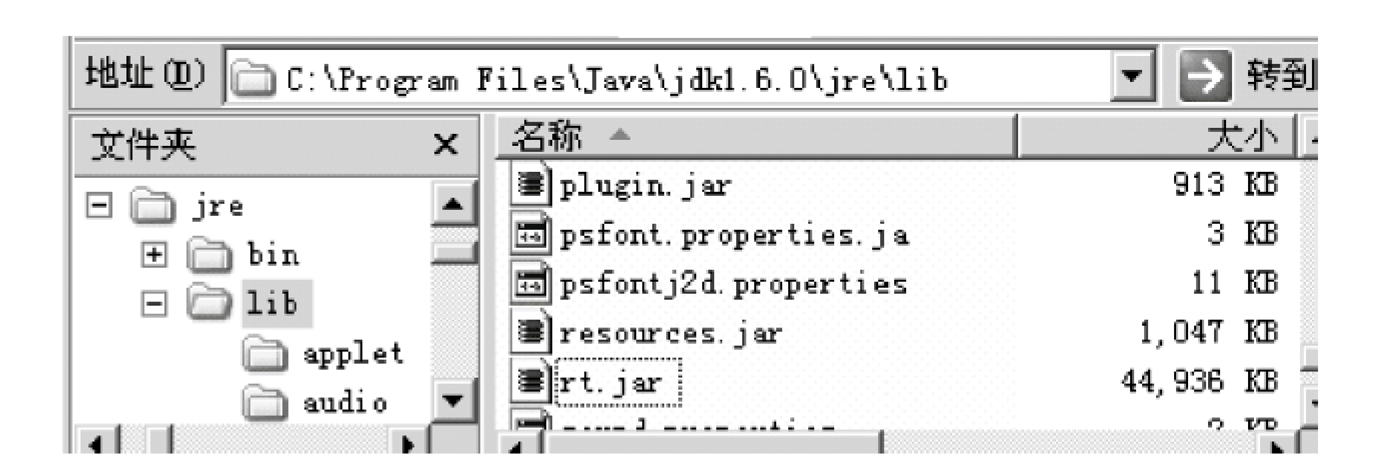


图 9.4　与 JAXP API 有关的系统库文件

2. JAXP API 中与 DOM 编程有关的软件包

(1) org.w3c.dom 软件包中的 API 介绍

在 org.w3c.dom 软件包中定义了实现 W3C 所制定的 DOM2 API 中的各个接口，其中核心接口中的成员方法和成员属性的功能说明请见本章中 9.1.3 小节“DOM 技术规范中的核心 API”的介绍。利用软件包 org.w3c.dom 中的接口可以创建、遍历和修改 DOM 树模型中的目标节点对象。

(2) javax.xml.parsers 软件包中的 API 介绍

它是 Sun 公司为 Java 平台所提供的与 XML 解析器有关的系统库，其中包含了 DOM 解析器和 SAX 解析器的具体实现。利用 javax.xml.parsers 软件包中的 DocumentBuilder 类和 DocumentBuilderFactory 类，可以读取目标 XML 文档文件并实现将 XML 文档转换成 DOM API 中的 Document 文档对象，如下面的代码片段示例所示。

```
DocumentBuilderFactory oneFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder oneBuilder = oneFactory.newDocumentBuilder();
Document domDocument = oneBuilder.parse(new File("someOneFile.xml"));
domDocument.normalize();
```

首先需要建立一个解析器工厂，然后利用这个工厂对象获得一个具体的 XML 解析器对象。之所以使用 DocumentBuilderFactory 类，主要目的是为了创建与具体 XML 解析器无关的应用程序(但如果要应用 SAX 解析技术，则采用 SAXParserFactory 工厂类)，因为 DocumentBuilderFactory 类中的静态方法 newInstance()被调用时，它自动从系统当前的 CLASSPATH 环境中加载某个 XML 解析器或者直接应用 J2SDK 中内置的 XML 解析器程序，最终达到所开发的 XML 应用程序与具体的 XML 解析器程序的实现无关性。

而 DocumentBuilder 类中的 parse()方法接收一个 XML 文档的文件名作为输入参数，并返回一个 Document 接口对象。由于返回的 Document 对象代表了目标 XML 文档的树模型，直接编程 Document 对象也就实现了对目标 XML 文档的操作访问，与解析器实现程序无关。

应用 Document 接口中的 normalize()方法可以去掉在 XML 文档中作为格式化内容的空白符而映射在 DOM 树模型中无用的 Text 类型的节点对象，因为在 XML 文档中空白符也会被作为对象映射在 DOM 树模型中。

3. javax. xml. transform 软件包中的 API 介绍

在 JAXP 1.0 版本中，并没有直接提供有关的功能类将内存中的 DOM 树模型的 Document 对象写回到 XML 文档文件中，需要借助其他软件包中的一些辅助功能类。而在 JAXP 1.1 版本中，引入了对 XSLT 的支持，实现对 XML 文档进行变换（Translation）后，得到一个新的文档结构，最后将新生成或者修改后的 DOM 树模型写回到指定的 XML 文件中。

在 javax. xml. transform 软件包中定义了用于 XSLT 转换功能的 API，其中的 TransformerFactory 工厂类可用于创建 Transformer 转换器对象实例，最终也同样使得具体的程序代码与特定的变换器实现程序无关。

（1）javax. xml. transform. dom 软件包中的 API 介绍

软件包 javax. xml. transform. dom 中的 DOMSource 类的主要功能是实现将内存中的 DOM 树模型中的 Document 对象指定为 XSLT 转换中的 DOM 数据源。

（2）javax. xml. transform. stream 软件包中的 API 介绍

软件包 javax. xml. transform. stream 中的 StreamResult 类代表转换结果的 I/O 输出流 OutputStream 对象。

利用 DOMSource 和 StreamSource 类可以将经过 XSLT 转换后的 DOM 文档对象写入到目标 XML 文件中。如下代码片段示例实现将 domDocument 对象所代表的 XML 文档数据写回到某个目标 XML 文件中。

```
TransformerFactory oneFactory = TransformerFactory.newInstance();
Transformer oneTransformer = oneFactory.newTransformer();
DOMSource domDocumentSource = new DOMSource(domDocument);
StreamResult domResult = new StreamResult(new File("someOneFile.xml"));
oneTransformer.transform(domDocumentSource, domResult);
```

Transformer 类中的 transfrom 方法接收两个参数：一个代表数据源的 DOMSource 类型的对象和一个代表输出目标的 StreamResult 类型的对象。最终把 DOM 文档内容输出到一个 I/O 输出流对象中，当这个 I/O 输出流对象是一个本地文件流时，DOM 树模型中的 XML 文档内容也就被写入到指定的 XML 文件中了。

4. 在校验模式下解析 XML 文档

（1）启用 XML 文档中声明的 DTD 校验 XML 文档

如果所要解析的目标 XML 文档内声明了 DOCTYPE（文档类型定义，DTD），并且希望在解析 XML 文档数据时依据 DTD 校验文档，则可以改变 DocumentBuilderFactory（DOM 方式）或者 SAXParserFactory（SAX 方式）对象中的 validating 属性启用根据 DTD 校验 XML 文档的功能，见如下代码片段：

```
DocumentBuilderFactory oneFactory = DocumentBuilderFactory.newInstance();
oneFactory.setValidating(true);
```

（2）启用 XML 文档中声明的 Schema 校验 XML 文档

在 javax. xml. validation 包中提供了支持 Schema 校验 XML 文档的 API，将校验 XML

文档的 API 和解析 XML 文档的 API 相互分离，如下方法中的代码示例也反映出这个特点。

```
public void validateXMLBySchema() throws Exception{
    DocumentBuilderFactory oneFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder oneBuilder = oneFactory.newDocumentBuilder();
    Document domDocument = oneBuilder.parse(new File("someOneFile.xml"));
    domDocument.normalize();
    SchemaFactory factory =                    //创建支持 W3C Schema 的工厂对象
    SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Source schemaFile = new StreamSource(new File("studentInfo.xsd"));  获得一个代表
    Schema schema = factory.newSchema(schemaFile);                      Schema 文件的
    Validator validator = schema.newValidator();                        Schema 对象
    try {                                      //对 DOM 树模型中的各个节点进行验证
        validator.validate(new DOMSource(domDocument));
    }                                          基于 Schema 对象
    catch (IOException e) {                    创建对应的验证
        // 验证失败时的错误处理代码,在此省略       器Validator对象
    }
    catch (SAXException e) {
        // 验证失败时的错误处理代码,在此省略
    }
}
```

应用 Schema 校验 XML 文档首先要创建 SchemaFactory 类的对象实例，并在 newInstance()方法的参数中提供一个 javax.xml.XMLConstants 类中定义的符号常量说明所应用的 Schema 文件的类型。本示例代码采用 XMLConstants.W3C_XML_SCHEMA_NS_URI，表示 Schema 文件是基于 W3C 规范的 XML Schema。

然后应用 SchemaFactory 对象中的 newSchema()方法创建一个包装目标 Schema 文件的 Schema 对象，再依据该 Schema 对象创建验证器 Validator 类的对象实例，最后就可以对内存中的 DOM 树模型中的各个节点对象进行验证了。

5. 创建 DOM 树模型和输出 Document 对象

(1) 利用 JAXP API 实现 DOM 编程的基本步骤

- 首先通过 DocumentBuilderFoctory 类创建一个 XML 解析器工厂对象，代码示例如下：

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
```

- 然后通过解析器工厂对象创建一个可以加载并生成 XML DOM 树模型的 DocumentBuilder 对象，代码示例如下：

```
DocumentBuilder builder = factory.newDocumentBuilder();
```

- 接下来，通过 DocumentBuilder 对象加载并生成 XML DOM 树模型中的 Document 对象的实例，代码示例如下：

```
            Element root = domDocument.getDocumentElement();            //获取文档的根元素
            domDocument.removeChild(root);
            return domDocument;
        }
        public void outPutDOMDocument(Document domDocument)
            throws TransformerException{
            TransformerFactory oneFactory = TransformerFactory.newInstance();
            Transformer oneTransformer = oneFactory.newTransformer();
            java.util.Properties properties =
                                 oneTransformer.getOutputProperties();
            properties.setProperty(OutputKeys.ENCODING,"gb2312");       //默认为 UTF-8
            properties.setProperty(OutputKeys.METHOD, "xml");
            properties.setProperty(OutputKeys.VERSION, "1.0");
            properties.setProperty(OutputKeys.INDENT, "yes");//换行,为 no 则不换行
            oneTransformer.setOutputProperties(properties);
            DOMSource domDocumentSource = new DOMSource(domDocument);
            StreamResult domResult   =
                           new StreamResult(new File("xmlDOMDataCopy.xml"));
    //如果要求在控制台中输出,应该采用如下的语句创建输出流对象
    //      StreamResult domResult = new StreamResult(System.out);
            oneTransformer.transform(domDocumentSource,domResult );
        }
        public static void main(String[] args)throws Exception    {
            Document domDocument = null;
            ModifyXmlByDOM oneModifyXmlByDOM = new ModifyXmlByDOM();
            domDocument = oneModifyXmlByDOM.createDOMDocument();
            domDocument = oneModifyXmlByDOM.updateDOMDocument(domDocument);
            oneModifyXmlByDOM.outPutDOMDocument(domDocument);
        }
    }
```

例 9-2 中的示例主要是演示如何将 XML 文件转换为 DOM 树模型,然后说明如何对 DOM 树模型中的节点进行修改操作(本示例是利用 Node 接口中的 removeChild()方法删除文档中的所有节点,最后产生一个空的 XML 文档),最后说明如何又将内存中的 DOM 树模型输出到目标 XML 文件中。该示例的执行结果如图 9.5 所示,最终是产生了一个空的 XML 文件,如果改变输出的 I/O 流对象的类型,还可以产生不同形式的输出结果。

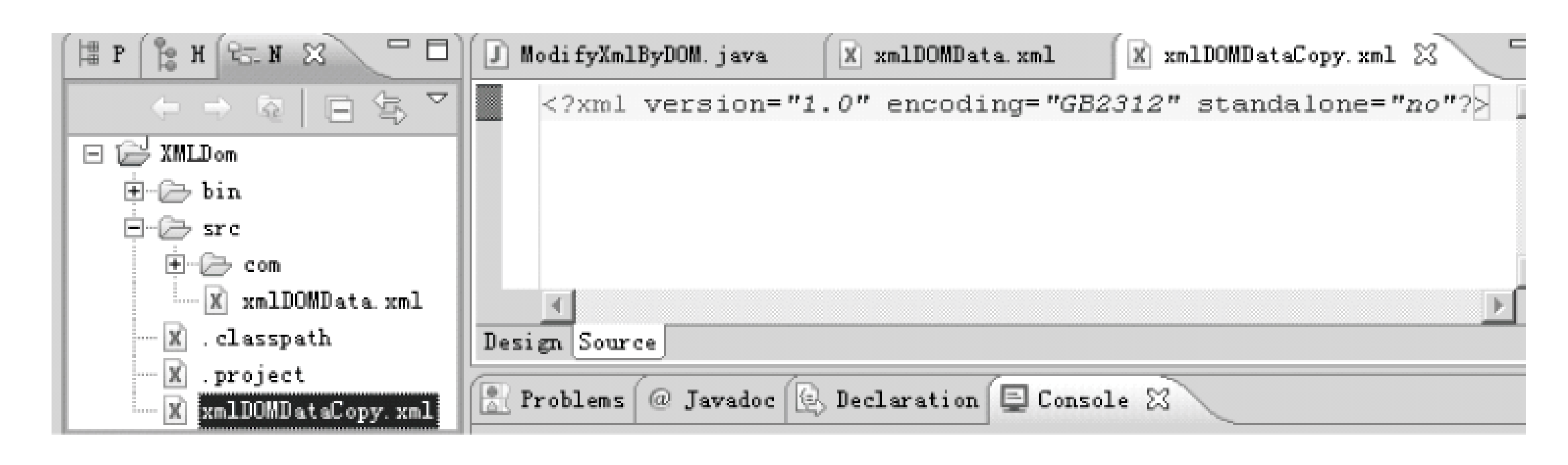


图 9.5 例 9-2 中示例代码的执行结果

如果将例 9-2 示例中的 StreamResult domResult 对象的创建代码改变为黑体所标识的语句,将可以在系统控制台中输出。当然,如果改变为 ServletOutputStream 流对象,将可

以直接向 Web 浏览器输出 XML 文档。

在创建 Document 对象时有可能抛出 ParserConfigurationException 异常，转换 Node 节点对象并生成 Transformer 对象时也有可能抛出 TransformerConfigurationException 异常，产生这些异常的主要原因一般是没有找到 XML 解析器程序。

9.2.2 在 Java 平台中应用 DOM 技术

1. 利用 DOM API 动态创建 XML 文档示例

例 9-3 为一个利用 DOM API 动态创建 XML 文档示例，在该示例中设计了 3 个方法。其中的 createDOMDocument()方法动态创建 XML 文档的 DOM 树模型，并获得一个代表空文档的 Document 对象，然后再为该文档提供一个定位 CSS 样式表 studentInfo. css 文件的处理器指令。

而 createAllElementsInDocument()方法在 DOM 树模型中的 Document 对象内动态地创建各个标签元素节点、子标签元素节点，以及添加标签元素的属性、文本对象、注释文本和 CDATA 段文本；outPutDOMDocument()方法将动态创建出的 Document 对象再写回到目标 XML 文件中。

例 9-3 利用 DOM API 动态创建 XML 文档代码示例

```
package com.px1987.webcrm.domparse;
import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.CDATASection;
import org.w3c.dom.Comment;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.ProcessingInstruction;
public class CreateXMLFileByDOM {
    public CreateXMLFileByDOM() {
    }
    public Document createDOMDocument() throws ParserConfigurationException{
        DocumentBuilderFactory oneFactory =
                DocumentBuilderFactory.newInstance();
        DocumentBuilder oneBuilder = oneFactory.newDocumentBuilder();
        Document domDocument = oneBuilder.newDocument();
        ProcessingInstruction pi =
```

```
            Document domDocument = null;
            CreateXMLFileByDOM oneCreateXMLFileByDOM = new CreateXMLFileByDOM();
            domDocument = oneCreateXMLFileByDOM.createDOMDocument();
            oneCreateXMLFileByDOM.createAllElementsInDocument(domDocument);
            oneCreateXMLFileByDOM.outPutDOMDocument(domDocument);
        }
    }
```

例 9-3 中的示例主要是演示如何动态创建 XML 文档，并在文档中创建特殊的节点，如属性对象、文本对象、注释文本和 CDATA 段文本对象等，但必须要应用对应的父节点对象中的 appendChild()方法添加到父节点对象中。该示例的执行结果如图 9.6 所示，所创建的 XML 文档类似于例 9-1 所示的示例文档。



图 9.6　例 9-3 中示例代码的执行结果

如果将例 9-3 示例中黑体所标识的创建 CSS 样式表文件的处理器指令改变为如下的 XSLT 样式表文件的代码，将能够对例 9-3 示例 XML 文档进行转换输出。

```
ProcessingInstruction pi =
            domDocument.createProcessingInstruction("xml - stylesheet",
            "type = \"text/xsl\" href = \"studentInfo.xsl\"");
```

2. 利用 DOM API 查询 XML 文档示例

(1) 本示例的功能要求

以例 7-10 中描述软件学院学生信息的 XML 文档为示例，设计一个利用 DOM API 查询该 XML 文档的程序。具体的功能要求如下：

- 统计出“学生信息”标签的个数(也就是有几个学生)，并显示输出统计的结果。
- 显示输出每个学生的各个方面的信息。

(2) 设计包装学生信息的实体组件 StudentInfoBean 类

为了避免每次进行数据访问请求时都重复地对 XML 文档进行解析，要缓存对例 7-10 示例解析后的结果。本示例采用一个实体类 StudentInfoBean 类进行包装，例 9-4 为包装解析结果的实体类 StudentInfoBean 的程序代码示例。

例 9-4　包装学生信息的实体组件 StudentInfoBean 类的代码示例

```
package com.px1987.webcrm.domparse;
public class StudentInfoBean {
```

```
private String name = null;
private String sex = null;
private String birthday = null;
private String majoy = null;
private String xmlCourseScore = null;
private String className = null;
private String cMajoy = null;
public String getXmlCourseScore() {
    return xmlCourseScore;
}
public void setXmlCourseScore(String xmlCourseScore) {
    this.xmlCourseScore = xmlCourseScore;
}
public StudentInfoBean() {
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getSex() {
    return sex;
}
public void setSex(String sex) {
    this.sex = sex;
}
public String getBirthday() {
    return birthday;
}
public void setBirthday(String birthday) {
    this.birthday = birthday;
}
public String getMajoy() {
    return majoy;
}
public void setMajoy(String majoy) {
    this.majoy = majoy;
}
public String getClassName() {
    return className;
}
public void setClassName(String className) {
    this.className = className;
}
public String getCMajoy() {
    return cMajoy;
}
public void setCMajoy(String majoy) {
```

```
            cMajoy = majoy;
        }
        public String toString(){
         return "姓名:" + name + "\t" + "性别:" + sex + "\t\t" + "出生日期:" + birthday + "\t" +
                "专业:" + majoy + "\n" + "班级:" + className + "\t" + "方向:" + cMajoy + "\t" +
                "XML 考试成绩:" + xmlCourseScore + "\n";
        }
    }
```

(3) 设计查询 XML 文档数据的 QueryXMLDataByDOM 类

例 9-5 为一个利用 DOM API 查询 XML 文档数据的 QueryXMLDataByDOM 类的代码示例,在该示例中设计了 2 个功能方法。其中的 createDOMDocument()方法动态加载例 7-10 示例 XML 文档,并获得一个代表该文档的 DOM 树模型中的 Document 对象。

而 queryTagDataInDocument()方法首先根据 XML 文档中各个节点的层次关系遍历每个<学生信息>节点,并获得每个<学生信息>节点中的"性别"属性和各个子节点中的文本。然后将每个学生信息包装到 StudentInfoBean 对象中,并依次添加到 List 集合中。最后将所获得的每个学生信息在控制台中打印输出。

例 9-5 查询 XML 文档数据的 QueryXMLDataByDOM 类代码示例

```
package com.px1987.webcrm.domparse;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
public class QueryXMLDataByDOM {
    public QueryXMLDataByDOM() {
    }
    public Document createDOMDocument() throws ParserConfigurationException,
                                            SAXException, IOException{
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        InputStream xmlInputStream = this.getClass().
                                  getResourceAsStream("/studentInfo.xml");
        Document domDocument = builder.parse(xmlInputStream);
        domDocument.normalize();
        return domDocument;
    }
public  List<StudentInfoBean> queryTagDataInDocument(Document domDocument){
    List<StudentInfoBean> allStudentInfos  = new ArrayList<StudentInfoBean>();
    Element rootElement  =  domDocument.getDocumentElement();  //取得根节点
```

```
        NodeList allStudentInfoNodeList =
        rootElement.getElementsByTagName("学生信息");        //取得"学生信息"节点集合
        mappingXMLElementToJavaBean(allStudentInfos,
                                    allStudentInfoNodeList );
        return allStudentInfos;
    }
    private  void mappingXMLElementToJavaBean(List<StudentInfoBean>
                    allStudentInfos,NodeList allStudentInfoNodeList ){
        StudentInfoBean oneStudentInfoBean = null;
    for(int index = 0; index < allStudentInfoNodeList.getLength(); ++index){
        oneStudentInfoBean = new StudentInfoBean();  //使用循环访问每一个学生信息节点
            Element oneStudentInfoNode =                //取得某个"学生信息"节点
                    (Element)allStudentInfoNodeList.item(index);
            String oneStudentSex = oneStudentInfoNode.getAttribute("性别");
            oneStudentInfoBean.setSex(oneStudentSex);       //取得"性别"属性的数据
            String studentName = oneStudentInfoNode.
                                    getElementsByTagName("姓名").
                                    item(0).getFirstChild().getNodeValue();
            oneStudentInfoBean.setName(studentName);
            String studentBirthday = oneStudentInfoNode.
                                    getElementsByTagName("出生日期").
                                    item(0).getFirstChild().getNodeValue();
            oneStudentInfoBean.setBirthday(studentBirthday);
            String studentMajoy = oneStudentInfoNode.
                                    getElementsByTagName("专业").
                                    item(0).getFirstChild().getNodeValue();
            oneStudentInfoBean.setMajoy(studentMajoy);
                                    Element classNameElement = (Element)oneStudentInfoNode.
                                    getElementsByTagName("班级").item(0);
            String studentClassName = classNameElement.      //获得"班级"节点的文本
                                    getFirstChild().getNodeValue();
            oneStudentInfoBean.setClassName(studentClassName);
            String studentCMajoy = classNameElement.getAttribute("方向");
            oneStudentInfoBean.setCMajoy(studentCMajoy);
            String xmlCourseScore = oneStudentInfoNode.
                                    getElementsByTagName("XML考试成绩").
                                    item(0).getFirstChild().getNodeValue();
            oneStudentInfoBean.setXmlCourseScore(xmlCourseScore);
            allStudentInfos.add(oneStudentInfoBean);
        }
        public static void main(String[] args) throws SAXException,
                        ParserConfigurationException,IOException {
        Document domDocument = null;
        List<StudentInfoBean> allStudentInfos = null;
        QueryXMLDataByDOM oneQueryXMLDataByDOM = new QueryXMLDataByDOM();
        domDocument = oneQueryXMLDataByDOM.createDOMDocument();
        allStudentInfos = oneQueryXMLDataByDOM.
                    queryTagDataInDocument(domDocument);
        System.out.println("学生信息有" + allStudentInfos.size() + "条\n");
```

应用例 9-6 示例中的 queryTagDataInDocument()方法的代码代替例 9-5 示例中的 queryTagDataInDocument()方法的代码，其他的方法代码不变，然后再执行例 9-5 示例，同样可以获得如图 9.7 所示的查询结果。

4. 利用 DOM API 动态修改 XML 文档示例

例 9-7 为一个利用 DOM API 动态修改 XML 文档数据的 UpdateXMLDataByDOM 类的代码示例，在该示例中设计了 3 个功能方法。其中的 createDOMDocument()方法动态加载例 7-10 示例 XML 文档，并获得一个代表该文档的 DOM 树模型中的 Document 对象。

而 updateTagDataInDocument ()方法修改<学生信息>节点中的指定元素值，本示例中的功能要求是将每个<学生信息>节点中的“性别”属性为“男”修改为“女”；然后利用 outPutDOMDocument()方法将动态修改后的 Document 对象再写回到目标 XML 文件中。为了方便检查程序执行的功能结果，将 I/O 输出流对象设置为 System. out，它代表系统控制台，并注意其中黑体所标识的语句。

例 9-7 利用 DOM API 动态修改 XML 文档的代码示例

```
package com.px1987.webcrm.domparse;
import java.io.IOException;
import java.io.InputStream;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
public class UpdateXMLDataByDOM {
    public UpdateXMLDataByDOM() {
    }
    public Document createDOMDocument() throws ParserConfigurationException,
    SAXException, IOException{
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        InputStream xmlInputStream =
            this.getClass().getResourceAsStream("/studentInfo.xml");
        Document domDocument = builder.parse(xmlInputStream);
        domDocument.normalize();
        return domDocument;
    }
    public  void updateTagDataInDocument(Document domDocument){
```

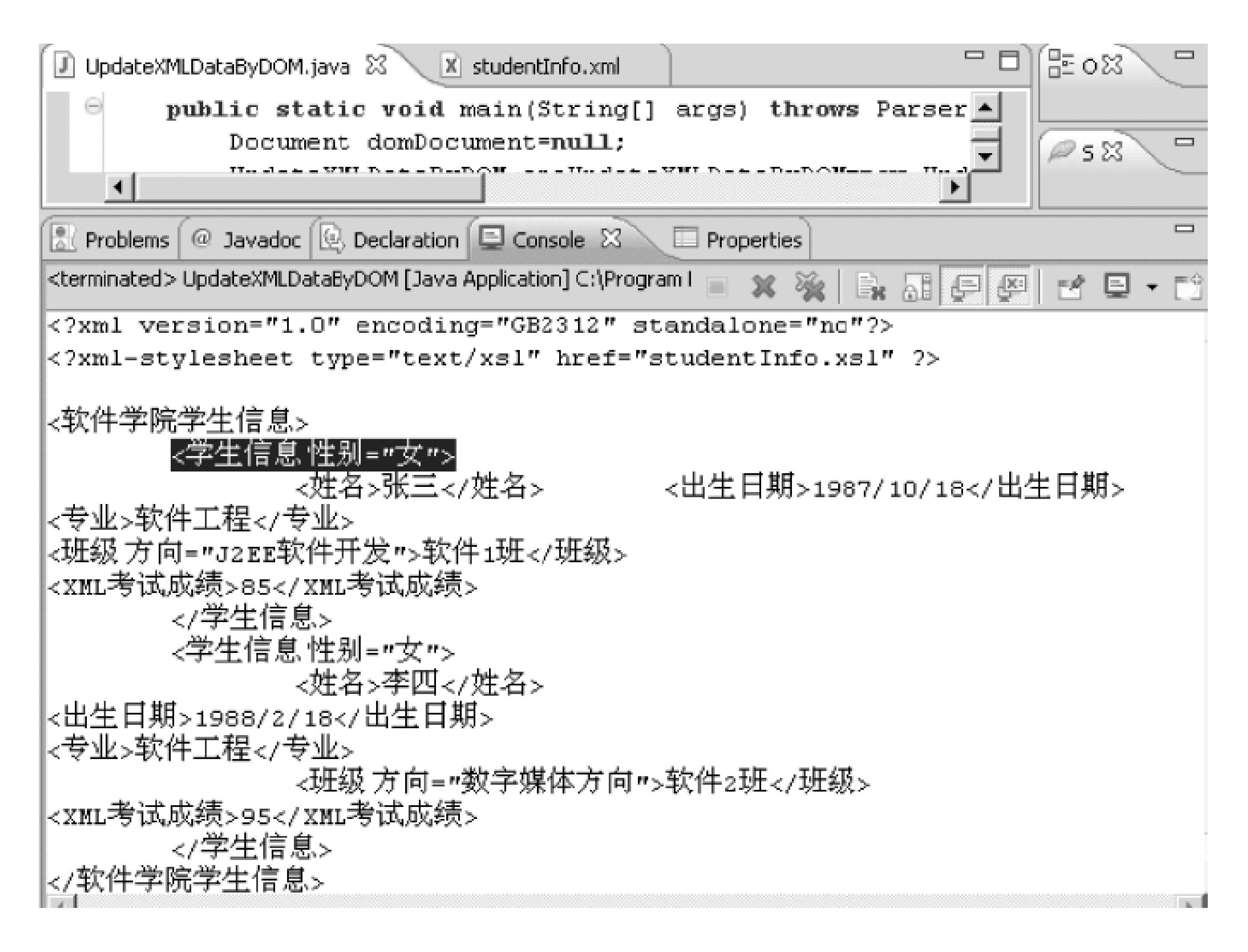


图 9.8 例 9-7 示例执行后在控制台中输出的结果

XML 文件中的数据写入到 Oracle 数据库表中。

在例 9-8 中，利用 DOM API 设计和编程实现了一个读取 MySQL 数据库表中代表银行卡信息的各条记录数据，并将它们保存到一个 XML 文件中的程序示例。读者只需要将其中的 connectToDB 方法替换为自己的 JDBC 连接 Connection 对象的获取程序代码，就能够操作指定的数据库系统中的数据库文件了。

例 9-8 根据数据库表中的数据创建 XML 文档的程序示例

```
package com.px1987.webcrm.domparse;
import java.io.File;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
public class GetDBDataToXMLByDOM {
    Connection oneConnection = null;
```

```
        }
    public static void main(String[ ] args) throws Exception {
        String targetFileName = "account.xml";
        String sqlStatement = "select * from bankcard";    //bankcard 为数据库表名
        GetDBDataToXMLByDOM oneGetDBDataToXMLByDOM = new GetDBDataToXMLByDOM();
        oneGetDBDataToXMLByDOM.connectToDB();
        oneGetDBDataToXMLByDOM.queryDBData(sqlStatement);
        oneGetDBDataToXMLByDOM.createDOMDocument();
        oneGetDBDataToXMLByDOM.createAllElementsInDocument();
        oneGetDBDataToXMLByDOM.outPutDOMDocument();
    }
}
```

在 Eclipse 开发工具中执行例 9-8 中的程序代码示例，将创建出如图 9.9 所示的 XML 文件，该 XML 文件中的各个标签数据与数据库表中的数据完全保持一致，已经成功地将银行卡信息数据库表中的各条记录都导出到 XML 文件中了。

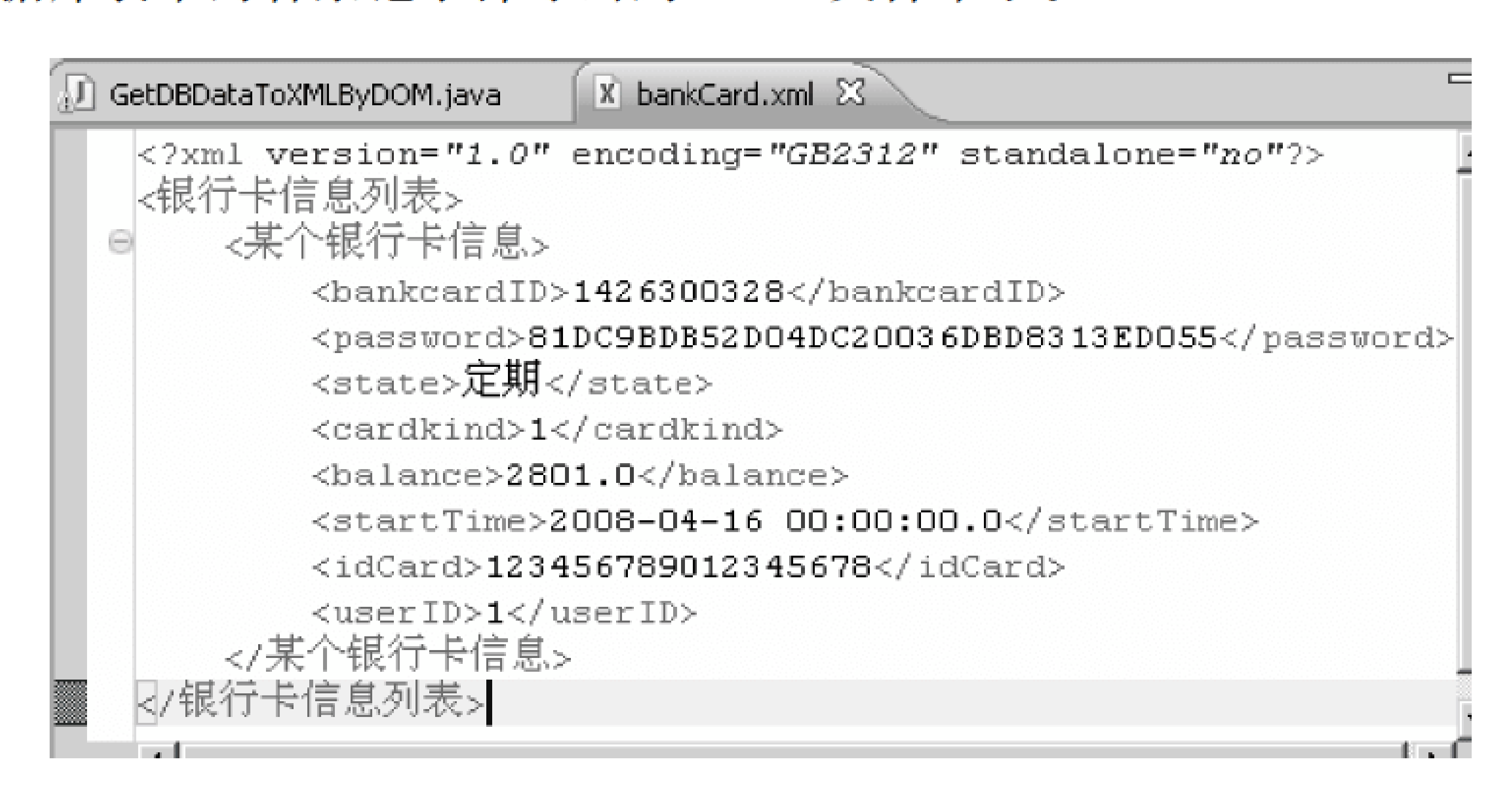


图 9.9 例 9-8 示例执行后的结果

如果再进一步将导出到 XML 文件中的数据写入到 Oracle 数据库表中，也就能够实现 MySQL 数据库表和 Oracle 数据库表之间进行数据交换了。

9.3 微软 IE浏览器对 XML DOM技术的扩展支持

9.3.1 HTML 页面中的 XML 数据岛技术

1. HTML 页面中的 XML 数据岛

(1) 什么是 XML 数据岛

数据岛就是被 HTML 页面引用或包含的 XML 文档中的数据，并且 XML 文档中的数据可以包含在 HTML 页面文件内，也可以包含在外部文件内，称为 HTML/XML 数据绑定技术。

```
                <author>张三</author>
                <publisher>清华大学出版社</publisher>
                <bookid>100</bookid>
                <price>35.5</price>
        </book>
        <book>  <title>Java 编程技术</title>
                <author>李四</author>
                <publisher>电子工业出版社</publisher>
                <bookid>101</bookid>
                <price>45.6</price>
        </book>
      </books>
    </xml>
</body><html>
```

4. 在页面的某个标签中应用 XML 数据岛中的数据

当将一个 HTML 标签元素绑定到一个 XML 标签元素时，HTML 标签元素会自动显示所绑定的 XML 标签元素的内容。例 9-11 中的<table>标签引用在例 9-10 示例中所声明的 XML 数据岛，其中的 datasrc 代表所声明的数据岛名称(但需要加“＃”字符)。

例 9-11　在页面的某个标签中应用 XML 数据岛中的数据的代码示例

```
<table datasrc="#xmlBooksID" border="1" cellpadding="5">
    <thead><th>书名</th><th>作者</th><th>出版社</th><th>书号</th><th>价格</th>
    </thead>
    <tr align="center">
       <td><span datafld="title" ></span></td>
       <td><span datafld="author"></span></td>
       <td><span datafld="publisher"></span></td>
       <td><span datafld="bookid"></span></td>
       <td><span datafld="price"></span></td>
    </tr>
</table>
```

最终将在<span>标签中显示 XML 文档中各个标签元素的数据，其中的 datafld 属性指定目标 XML 标签名。

5. 在 HTML 页面中采用链接外部的 XML 文档方式应用数据岛示例

例 9-12 为一个在 HTML 页面中链接外部的 XML 文档的代码示例，并在该页面中设计了一个表格，包括表头和表体两部分。在表格单元格中显示 XML 文档中的目标标签中的数据。

例 9-12　在 HTML 页面中链接外部的 XML 文档的代码示例

```
<html><head><title>将 XML 中的数据绑定到 HTML 页面中</title></head><body>
    <xml id="xmlBooksID" src="Books.xml"></xml>
    <table datasrc="#xmlBooksID" border="1" cellpadding="5">
```

```
        <thead><th>书名</th><th>作者</th><th>出版社</th><th>书号</th><th>价格</th>
        </thead>
            <tr align="center">
                <td><span datafld="title" ></span></td>
                <td><span datafld="author"></span></td>
                <td><span datafld="publisher"></span></td>
                <td><span datafld="bookid"></span></td>
                <td><span datafld="price"></span></td>
            </tr>
        </table>
</body><html>
```

例 9-13 为例 9-12 示例页面所绑定的目标 XML 文档中的标签数据示例，XML 文件名为“Books. xml”。

例 9-13　XML 文件 Books. xml 中的代码示例

```
<?xml version="1.0" encoding="GB2312"?>
<books>
    <book><title>XML 编程技术</title>
            <author>张三</author>
            <publisher>清华大学出版社</publisher>
            <bookid>100</bookid>
            <price>35.5</price>
    </book>
    <book><title>Java 编程技术</title>
            <author>李四</author>
            <publisher>电子工业出版社</publisher>
            <bookid>101</bookid>
            <price>45.6</price>
    </book>
</books>
```

例 9-12 示例页面在 IE 浏览器中执行的结果如图 9. 10 所示，获得了 XML 文档中的全部数据，并且以表格形式显示输出。



图 9. 10　例 9-12 示例页面在 IE 浏览器中执行的结果

6. 在 HTML 页面中采用内部嵌入 XML 文档方式应用数据岛示例

例 9-14 为一个应用内部嵌入 XML 文档方式使用数据岛的代码示例。由于已经将 XML 文档完整地内嵌到 HTML 页面中，因此不再需要外部的 XML 文档。然后在表格中

windows 平台上首个公用的 DOM 实现而开发的，并作为一个 ActiveX 控件。在 JavaScript 语言中引入了用于创建 ActiveX 对象的 ActiveXObject 类。

ActiveXObject 类的构造函数只有一个参数，代表要进行实例化的 ActiveX 对象的字符串代号。例如，XML DOM 对象的第一个版本的代号为“Microsoft.XMLDOM”。

3. 在 IE 浏览器中创建 XML DOM 对象

下面的代码片段示例实现在 IE 浏览器中创建 XML DOM 对象。

```
var xmlDocument = new ActiveXObject("Microsoft.XMLDOM");  //创建 XML 文档对象
xmlDocument.async = false;                               //关闭异步加载而采用同步加载方式
xmlDocument.load("XMLData.xml");                          //解析器加载 XML 文档
```

其中的 load()方法用于加载 XML 文件，执行这段代码后将创建出代表 XML 文档的 xmlDocument 对象，它与其他的 DOM Document 对象的行为完全一样。但如果以同步模式载入 XML 文件，JavaScript 代码将会等待 XML 文件完全载入完毕后才会继续执行；而以异步模式载入 XML 文件时，不会出现等待，但可以在事件处理函数中判断 XML 文件的内容是否完全载入完毕。

4. 在 JavaScript 脚本代码中动态装载 XML 文档

例 9-15 为一个在 JavaScript 脚本代码中动态装载 XML 文档的代码示例，其中设计了两个函数。createXMLDocument()函数首先创建一个解析 XML 文件的 ActiveX 组件对象，然后再动态加载目标 XML 文档，最终获得一个代表 XML 文档的 Document 对象；而 showXMLDataInHtmlTag()函数则检索出 XML 文档中的第一个<book>标签节点，然后再分别获得目标标签数据，并在 HTML 页面的标签中显示输出。

例 9-15 在 JavaScript 代码中动态装载 XML 文档的代码示例

```
<html><head><script language="JavaScript">
     var xmlDocument = null;
     function createXMLDocument(){
        xmlDocument = new ActiveXObject("Microsoft.XMLDOM");
        xmlDocument.async = false;
        xmlDocument.load("Books.xml");
     }
     function showXMLDataInHtmlTag(){
         var firstBookInfoNode =                          //取得第一个“book”节点
                 xmlDocument.getElementsByTagName("book")[0].childNodes;
         title.innerText = firstBookInfoNode.item(0).text;
         author.innerText = firstBookInfoNode.item(1).text;
         publisher.innerText = firstBookInfoNode.item(2).text;
         bookid.innerText = firstBookInfoNode.item(3).text;
         price.innerText = firstBookInfoNode.item(4).text;
     }
</script>
<title>在 HTML 页面中利用 JavaScript 调用 XML 文件中的数据</title></head>
```

```
<body>
    <script language="JavaScript" for="window" event="onload">
        createXMLDocument();
        showXMLDataInHtmlTag();
    </script>
    <b>标题: </b><span id="title"></span>br>
    <b>作者: </b><span id="author"></span><br>
    <b>出版社: </b><span id="publisher"></span><br>
    <b>书号: </b><span id="bookid"></span><br>
    <b>价格: </b><span id="price"></span>
</body></html>
```

例 9-15 所应用的 XML 文件“Books.xml”仍然为例 9-13 中的 XML 文档内容,其中的<script language="JavaScript" for="window" event="onload">是指当页面所在的浏览器窗口被创建和 HTML 页面载入时,将执行<script>标签内的 JavaScript 脚本代码。例 9-15 示例页面在 IE 浏览器中的最终执行的结果如图 9.11 所示。

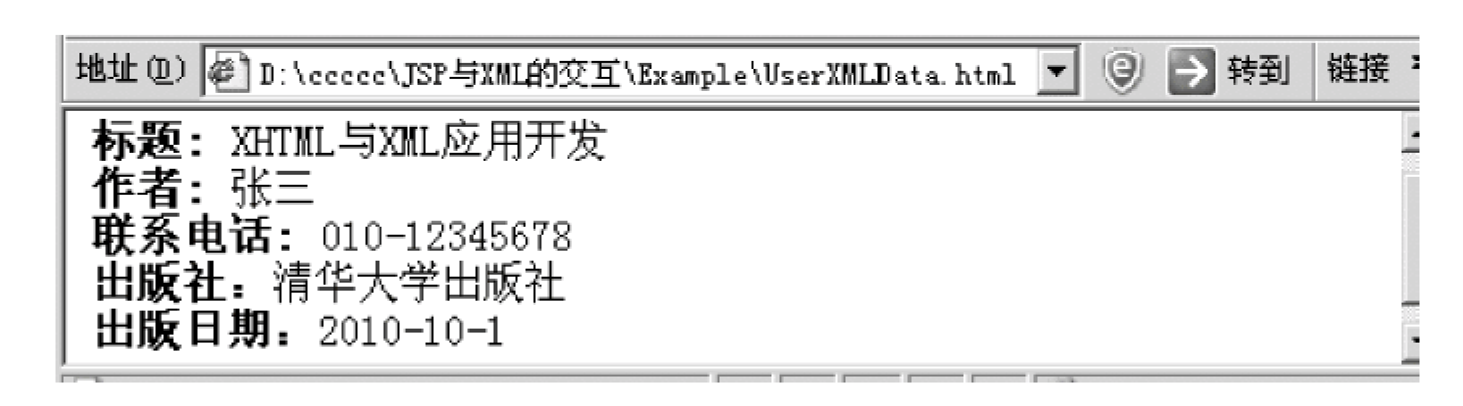


图 9.11 例 9-15 示例页面在 IE 浏览器中执行的结果

本章小结

教学重点

DOM 在支持 HTML 的基础上提供了一系列的扩展 API 以支持针对 XML 文档中的数据访问和操作。利用这些 API,开发人员可以很容易地从 XML 中提取信息,修改原始的信息,甚至包括重新构造新的 XML 文档。但由于 DOM 是基于树模型的转换机制,对大型的 XML 文档的解析应用 DOM 技术,则会影响到系统的总体效率。因此,在本章的教学中一定要让学生正确地了解 DOM 技术本身的优缺点和适用的应用场合。

DOM 本身有一套丰富的 API,而且是与语言无关的。因此,在教学中需要重点地讲解这些 API 及具体的应用。当然可以结合某种技术平台的实现给出具体的示例,本章主要是以 Java 语言为原型说明 DOM API 的具体应用的。

学习难点

DOM API 在具体编程应用时显得比较“繁杂”,这也为初学者带来一定的复杂性和难度。为此,开源的 JDom 把 SAX 和 DOM 的 XML 解析技术及功能实现有效地结合在一起,

尽管也继续应用树形结构转换 XML 文档中的各个节点，但在设计方面则充分利用了 Java 平台中的各种编程技术，如方法的重载、垃圾回收机制和后台线程处理技术等方面的技术支持。

因此，JDom 在系统 API 类库的设计方面简单和易用，并且这些 API 中的各个功能方法都采用 Java 程序员比较熟悉的形式来定义和描述 XML 文档中的各个标签。

在解析 XML 文档时如果需要启用其中的 DTD 或 Schema 验证功能，它们在具体的实现方式上是不同的。在 JAXP 中对于应用 DTD 验证，是将验证和解析合二为一；而对于启用 Schema 验证功能，则在代码实现方面是将解析和验证相互分离。在编程中，需要理解其中的差别。

教学要点

DOM 树模型中节点类型的划分规则和 XPath 中节点类型的划分规则是不同的，DOM 是基于“一切都是节点，节点都是对象”的设计原则，将 XML 文档中的所有不同类型的数据都转换为对应类型的对象。因此，在 DOM 树模型中共定义了 12 种不同类型的节点，这与 XPath 中的 7 种类型的节点是不同的。在教学中需要与 XPath 对比讲解，说明它们两者在节点类型划分方面的差别。

DOM 树模型中的根节点由 Document 表示，Document 代表整个 HTML/XML 文档内容本身，文档中的其他类型的元素所对应的节点都是 Document 节点的子节点。在教学中需要理顺 DOM 树模型中的各个节点和 XML 文档中元素之间的对应关系。

在教学中还需要分清标签元素节点和文本节点之间的不同，尽管文本节点可以作为标签元素节点的子节点，但文本节点并不是标签元素节点的值。本章重点介绍了在 Java 平台中对 W3C DOM 的技术实现和具体编程应用，但 DOM 本身是与编程语言无关的。因此，本章所介绍的内容，同样也适用于微软的 VS. NET 平台和浏览器中的 JavaScript 脚本语言。

学习要点

由于 DOM 规范在设计时追求通用性，以及在多种不同的编程语言中都可以实现 XML 解析程序，因此 DOM 是重量级的树模型解析实现技术。为此，在 Java 平台中相继提出了许多轻量级的 XML 解析实现技术。比如，基于事件流的 SAX 技术、基于树模型的 JDom 和 Dom4J 技术等。在应用开发中可以根据项目的需要选择其中的某种 XML 解析实现技术。

JDom 是轻量级的树模型解析实现技术，在对 XML 解析的性能方面要比 DOM 优越。关于 JDom 的具体编程及应用技术，作者在《J2EE 课程设计——技术应用指导》一书（见本书的参考文献）的第 6 章“XML 解析技术及在项目开发中的应用”中有详细介绍。

在 Java 程序中将指定的 XML 文档转换为 DOM 树模型，首先需要建立一个解析器工厂类对象实例，然后再利用这个工厂对象获得一个具体的 XML 解析器对象。其目的是为了能够创建与具体的 XML 解析器无关的应用程序，请仔细阅读例 9-2 中的代码示例。

参 考 文 献

［1］ 杨少波．J2EE 项目实训——Hibernate 框架技术．北京：清华大学出版社，2008
［2］ 杨少波．J2EE 项目实训——Spring 框架技术．北京：清华大学出版社，2008
［3］ 杨少波．J2EE 项目实训——Struts 框架技术．北京：清华大学出版社，2008
［4］ 杨少波．J2EE 项目实训——UML 及设计模式．北京：清华大学出版社，2008
［5］ 杨少波．J2EE 课程设计——项目开发指导．北京：清华大学出版社，2009
［6］ 杨少波．J2EE 课程设计——技术应用指导．北京：清华大学出版社，2009